2018

Sliding on a Massive Spinning Asteroid: Order and Chaos

Hannah Peltz Smalley
hpeltzsm@wellesley.edu

Sliding on a Massive Spinning Asteroid: Order and Chaos

Hannah E. Peltz Smalley
Physics Department
Wellesley College

A dissertation submitted in partial fulfillment of the requirements
of the Honors Thesis at Wellesley College

May 29, 2018

Project Adviser
John F. Lindner

Major Adviser
Robbie Berg

Department Chair
Glenn Stark

Reviewer
Jerome Fung

Honors Visitor
Barbara Geller

Abstract

In 2017 an interstellar visitor, 'Oumuamua, was discovered by Robert Weryk at Haleakala Observatory, Hawai'i. Analysis of 'Oumuamua's light curve suggests it has a highly elongated shape. The dynamics of such irregularly shaped objects cannot be modeled with simple approximations; while Newton's sphere theorem allows us to model the gravitational dynamics of spherically symmetric objects as point masses, a spherical approximation is insufficient to describe the interactions of highly elliptical objects like 'Oumuamua – and even Earth, which is not a true sphere but rather an oblate spheroid.

Fortunately, there exist closed-form expressions for the gravitational potential of a spheroid. In this project, I use *Mathematica* to solve systems of nonlinear differential equations to model the motion of an object sliding on such a spheroid. Under certain conditions, the path of the slider becomes highly sensitive to initial conditions. This sensitivity manifests as chaos, which is apparent both qualitatively and quantitatively.

Acknowledgments

I would like to express my gratitude to my adviser, Dr. John Lindner, who first thought to ask what geodesics on a spinning ellipsoid would look like, and without whose support this project would never have been possible. I would additionally like to thank the professors in the physics department at Wellesley College, in particular Dr. Robbie Berg, who worked with me to coordinate a thesis advised outside the department, and also taught me the Lagrangian method in the first place.

Completing my thesis would not have been possible without the impeccable parenting skills of my mother and father, who never once tried to talk me out of it, and lifelong support from my best friend Emily Panganiban, who forgives me every time I leave her on read for two weeks because I got distracted by a project.

Contents

Abstract **iii**

Acknowledgments **v**

1 Introduction **1**
1.1 Irregular Bodies in the Solar System 1
1.2 Geodesics 3
1.3 Chaos 3
1.3.1 The Lyapunov Exponents 4
1.3.2 Visualizing Chaos 5
1.4 Gravity 6

2 The Initial Value Problem **7**
2.1 The Lagrange Method: Examples 7
2.1.1 Simple Harmonic Oscillator 7
2.1.2 Motion on the Surface of a Static Sphere 8
2.2 Lagrangian of the Spheroid-Slider System 10
2.2.1 Viscosity 13
2.3 Nondimensionalization 13
2.4 Initial Value Problem 14

3 Chaos **17**
3.1 Detecting Chaos 17
3.1.1 Poincaré Sections 17
3.1.2 Lyapunov Exponent Calculation 19
3.2 The Maximum Lyapunov Exponent of the Lorenz System 24
3.3 Fractals 25
3.4 Numerical Integration and Accuracy of Results 25

4 Results **27**
4.1 Variable Lyapunov Exponents 27
4.1.1 Relationship to the Trace of the Jacobian 27
4.1.2 Results from the Graphical Lyapunov Exponent Algorithm 27
4.2 Poincaré Sections 31

4.2.1 Poincaré Sections With Viscosity 31
4.3 Conditions for Chaos versus Periodicity 31
4.4 Fractal Basins 32
4.5 Regions of Chaos 35

5 Conclusions **45**
5.1 Discussion 45
5.2 Challenges and Limitations 46
5.3 Future Work 47

Appendices **47**

A Dictionary of Symbols **49**

B Mathematica Notebooks **51**

Bibliography **81**

List of Figures

1.1 'Oumuamua brightness variation 1
1.2 433 Eros 2
1.3 Rollercoaster 3
1.4 Geodesics on an oblate spheroid 4
1.5 Sensitivity to initial conditions as demonstrated by Nathaniel Moore. 5

2.1 Schematic of the coordinate system on the sphere 8
2.2 Schematic of the coordinate system used to parametrize the equations of motion. 10
2.3 The effect of gravity on a slider on a highly eccentric ellipsoid. . 12
2.4 Zipper trajectory 13
2.5 Contour plots of the potential surface 14

3.1 Poincaré section of a physical forced damped pendulum 18
3.2 Poincaré section of a periodic orbit 19
3.3 Poincaré sections of an undamped driven pendulum 21
3.4 Comparison of methods used to calculate Lyapunov exponents for the undamped pendulum 23
3.5 Maximum Lyapunov exponent of the Lorenz system 24
3.6 The first four iterations of the Koch snowflake. 25
3.7 Comparison between orbits computed at high and low precision . 26
3.8 Comparison of Poincaré sections of orbits calculated with different working precision 26

4.1 Maximum Lyapunov exponent algorithm as applied to the spheroid-slider system 28
4.2 Nonlinear scaling regime for a chaotic orbit 30
4.3 Comparison of Poincaré sections for chaotic and periodic orbits . 31
4.4 The trace of the Jacobian 32
4.5 Poincaré sections with viscosity 33
4.6 Orbits with varied Ω around the axis of symmetry 34
4.7 Fractal basin plots for a massless oblate spheroid 35
4.8 Fractal basin plots for a prolate spheroid with potential surface . 36

4.9 Effect of viscosity on fractal basin plots 36
4.10 Effect of initial velocity on fractal basin plots 37
4.11 Comparison of the fractal basins for massless and massive prolate spheroids 38
4.12 Effect of initial launch angle on fractal basin plots 38
4.13 Sensitive regions on the spheroid 40
4.14 The divergence plots reflect the fractal basins for a massless spheroid 41
4.15 Divergence plots for a massive spheroid 43

Chapter 1

Introduction

1.1 Irregular Bodies in the Solar System

On 19 October 2017, 'Oumuamua, formally designated 1I/2017 U1, was discovered by Robert Weryk at Haleakala Observatory, Hawai'i [1]. The object is on a hyperbolic orbit through the solar system. Analysis of 'Oumuamua's light curve (Fig. 1.1) suggests it has a highly elongated shape.

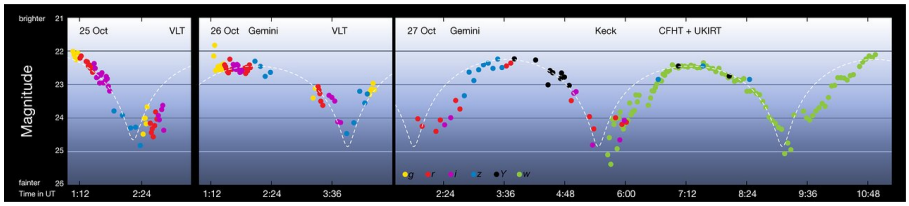


Figure 1.1: Variation in brightness of 'Oumuamua over the course of three days in October 2017. The large range of brightness suggests the object has a highly elongated shape. The dotted line represents what the light curve would look like if Oumuamua were an ellipsoid with a 1:10 ratio between the axes. By ESO/K. Meech et al [2].

The solar system is full of irregularly shaped objects, such as the asteroid Eros (Fig. 1.2), though most are not quite as eccentric as 'Oumuamua. In the chapters that follow, I use *Mathematica* simulations to explore what could happen to an object on the surface of such an eccentric object. It turns out that rotation gives rise to chaotic orbits that are highly sensitive to initial conditions, a phenomenon which is not seen on static ellipsoids.

In reality, an object on the surface of an asteroid–especially a not so massive one–would be in danger of being flung off. In our model system, however, the slider is always confined to the surface, like a car on a rollercoaster.

Figure 1.2: 433 Eros is a well-known asteroid with a peculiar saddle shape. A mosaic of six images taken February 29, 2000, by the NEAR spacecraft, from an orbital altitude of about 200 kilometers.

Figure 1.3: Like the cars on a rollercoaster, the slider is confined to the surface of the "asteroid."

The curved geometry of the spheroid-slider system makes the motion of the slider very sensitive to initial conditions. When the slider is released from rest, it is subject to pseudoforces in the spheroid reference frame. The centrifugal and coriolis effects on the slider, together with the variable curvature of the surface, gives rise to complicated dynamics.

1.2 Geodesics

Left to its own devices, a slider on a curved surface will follow the geodesic. There are several ways of characterizing geodesics. The simplest way is to define a geodesic as the shortest path between two local points. However, it is more general to define them as lines of zero geodesic curvature; or the equivalent of straight lines on a flat plane. Such a path parallel transports its tangent vector; that is, vectors tangent to the curve are parallel at all points along the curve.

Geodesics on the sphere are closed great circles. On the spheroid, however, more complicated trajectories are seen, as in Fig 1.4. The path of the geodesic does not return to the same spot after one revolution, and a large area of the spheroid may be covered.

A point sliding on the surface of such a spheroid is inclined to follow the geodesic, according to the principle of least action. So, a point launched at some initial velocity on a static spheroid subject to no other forces will simply follow the geodesic. There is a large body of mathematical work describing geodesics on static spheroids, dating back to the 19th century. However, to my knowledge none of this work deals with what happens when the spheroid is rotated, which would have been an intractable problem prior to modern computational techniques.

1.3 Chaos

In the absence of rotation, the slider's path may cover a large section of the surface, but it will leave behind an orderly pattern that repeats itself over many cycles. This pattern will be common to local trajectories and resistant to small perturbations, such as numerical errors. Rotating the spheroid about one of its axes, however, introduces pseudoforces that perturb the slider and sensitize it to its initial conditions. This sensitivity gives rise to chaotic behavior.

Chaos occurs in a system wherever there is a high sensitivity to initial conditions; that is, the slightest difference in the starting point will result in vastly different outcomes. Local trajectories will not appear similar and small errors

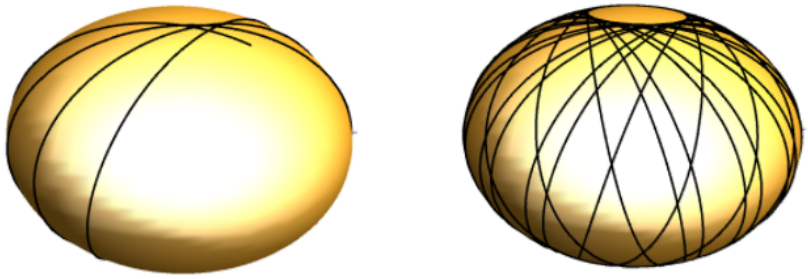


Figure 1.4: Geodesics on the ellipsoid do not close. Left shows the geodesic on an oblate spheroid after only a few circuits. Right shows the same geodesic after many circuits.

will confound long-term prediction. Such sensitivity is seen in systems that have at least three degrees of freedom, from forced damped pendulums to weather systems.

The spheroid-slider system was first studied in the summer of 2016 by Nathaniel Moore and John Lindner. Qualitative indicators of chaos were seen in the ellipsoid-slider system early on. Comparisons between different initial parameters show that for some parameters, local trajectories rapidly diverge, as in Fig. 1.5. The finding motivated this project to develop more quantitative methods of measuring chaos.

1.3.1 The Lyapunov Exponents

One metric that is often used to determine whether or not a system is chaotic is its spectrum of Lyapunov exponents. A Lyapunov exponent describes the rate at which an arbitrary trajectory of the system diverges from trajectories that have nearly the same initial conditions – that is, they start in almost the same place. A positive exponent indicates exponential divergence, which is the definition of chaos, while a negative exponent indicates that local trajectories will converge on the same behavior.

There is one exponent associated with each direction in the phase space. The phase space of the ellipsoid-slider system is five-dimensional; two dimensions for longitude and latitude, an additional two for their associated velocities, and one for the rotation phase. The set of exponents that describe the divergence in each direction is called the spectrum of Lyapunov exponents, the maximum of which will dominate the behavior of the system.

Calculating Lyapunov exponents is not straightforward and requires computational methods, as described in Chapter 3.

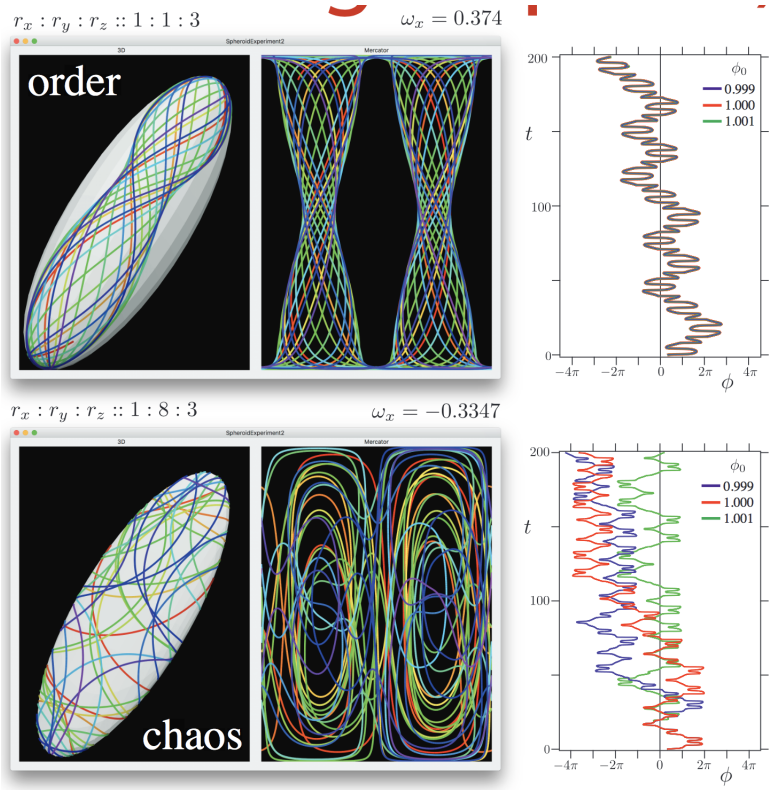


Figure 1.5: Left: Obj-C simulations of trajectories on two sample ellipsoids by Nathaniel Moore in 2016 [3]. One trajectory traces a consistent and orderly path, while the other wanders chaotically. Right: plots of time versus longitude for the two trajectories. The initial value of ϕ is perturbed by 0.001 units in both the positive and negative directions. The orderly trajectory is not perceptibly changed, but the chaotic trajectory diverges in ϕ.

1.3.2 Visualizing Chaos

The system of ellipsoid and slider is interesting in that it is a two-body system containing one body that is not a point particle. Chaos requires a phase space of at least three dimensions, so most two-body systems commonly studied in physics do not exhibit chaos because they are confined to a two-dimensional plane. In the spheroid-slider system, however, the slider may move in three dimensions. The system is also unusual in that positive curvatures are often associated with order, and hyperbolic surfaces are described as chaotic. The ellipsoid is positively curved, but through rotation gives rise to chaos.

Poincaré Sections

Poincaré sections can exhibit qualitative indicators of chaos. Each section is a plot of some combination of parameters represented by points in phase space. Chaotic regions of a section contain many points spread out over a large area, while periodic regions form continuous lines, as seen in Fig. 3.3. In some systems, the presence of a strange attractor will create fractal geometries in the phase space, such as the section in Fig 3.1.

Fractals

A fractal is a shape with fractional (non-integer) dimensions, which results in a self-similar pattern; the shape looks nontrivially the same no matter how much it is magnified. Fractals are associated with dissipative systems, and are only seen in the spheroid-slider system when a viscous frictional parameter is added. Fractal patterns emerge in basin plots of the spheroid-slider system.

1.4 Gravity

Introducing gravity changes the potential surface of the spheroid, which previously was completely determined by the centrifugal pseudoforce. Gravity is simulated by calculating the potential by the method outlined in MacMillan's Theory of the Potential [4], and then incorporating that potential into the Euler-Lagrange method used to derive the equations of motion. The effect of gravity is not required in order to observe interesting trajectories, but its presence makes the system more physically relevant.

The gravitational potential and the centrifugal potential create "hills" and "valleys" that affect the motion of the slider. In cases where the two contributions to the total potential are similar in magnitude, multiple hills and valleys can be observed across the surface.

Chapter 2

The Initial Value Problem

2.1 The Lagrange Method: Examples

Before constructing the full generalized initial value problem of a sliding point on a massive rotating spheroid, I will demonstrate the utility of the Lagrange method with simpler cases.

2.1.1 Simple Harmonic Oscillator

Consider a mass m on a spring of constant k. By Hooke's Law, the force on the mass due to the spring is

$$F = -kx, \tag{2.1}$$

where x is the position of the mass along the x-axis.

Then the potential energy is the integral of F

$$U = -\int F dx = \frac{1}{2}kx^2 \tag{2.2}$$

and if the velocity is simply the time derivative of the position x, the kinetic energy is

$$T = \frac{1}{2}m\dot{x}^2. \tag{2.3}$$

As detailed in Taylor's *Classical Mechanics* [5], the Lagrangian is defined as $L = T - U$. So

$$L = \frac{1}{2}(m\dot{x}^2 - kx^2) \tag{2.4}$$

which implies one Euler-Lagrange equation

$$\frac{\partial L}{\partial x} = \frac{d}{dt}\frac{\partial L}{\partial \dot{x}}. \tag{2.5}$$

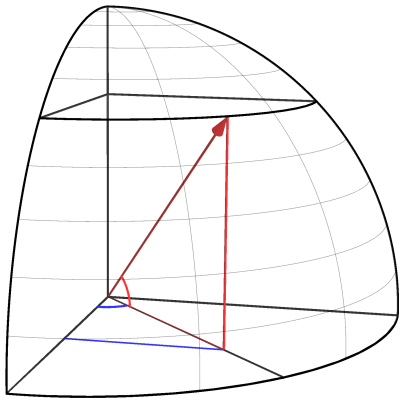


Figure 2.1: An octant of the sphere showing the latitudinal angle λ in red and the longitudinal angle ϕ in blue.

Calculating the prescribed derivatives on L yields

$$\frac{\partial L}{\partial x} = -kx \tag{2.6}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = m\ddot{x} \tag{2.7}$$

and plugging these into Eq. 2.26 yields

$$m\ddot{x} + kx = 0. \tag{2.8}$$

In the case of oscillations on a spring, the constant k can be expressed in terms of the attached mass m and the angular frequency of the motion ω as $k = \omega^2 m$, which reduces Eq. 2.8 to

$$\ddot{x} + \omega^2 x = 0. \tag{2.9}$$

The solution to this differential equation is sinusoidal,

$$x(t) = \sin \omega t + \varphi. \tag{2.10}$$

2.1.2 Motion on the Surface of a Static Sphere

Take the simple case of a static massless sphere. (It turns out this case is identical to the massive sphere, because the gravitational potential is the same everywhere.)

We define a point on the ellipsoid using a triplet of coordinates $\{x, y, z\}$, but we put these coordinates in terms of the angles λ and ϕ with respect to the center of mass of the ellipsoid, as shown in Fig. 2.1. So the position vector, the equivalent of the 1-dimensional x position in the previous case, looks like

$$\vec{r}(t) = \{\cos\lambda(t)\cos\phi(t), \cos\lambda(t)\sin\phi(t), \sin\lambda(t)\} \tag{2.11}$$

and to get the velocity take the time derivative, like so

$$\dot{\vec{r}} = \{-\dot{\lambda}\sin\lambda\cos\phi - \dot{\phi}\cos\lambda\sin\phi, \dot{\phi}\cos\lambda\cos\phi - \dot{\lambda}\sin\lambda\sin\phi, \lambda\cos\lambda\}. \tag{2.12}$$

Taking $U = 0$, the Lagrangian is simply

$$L = \frac{1}{2}m\dot{\vec{r}}\cdot\dot{\vec{r}} = \frac{1}{2}m\left(\dot{\lambda}^2 + \dot{\phi}^2\cos^2\lambda\right), \tag{2.13}$$

where m is the mass of the slider.

In this case, we have two Euler-Lagrange equations,

$$\frac{\partial L}{\partial \lambda} = \frac{d}{dt}\frac{\partial L}{\partial \dot{\lambda}} \tag{2.14}$$

and

$$\frac{\partial L}{\partial \phi} = \frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}}. \tag{2.15}$$

We can evaluate the equations just as before. They are

$$-\dot{\phi}^2\sin\lambda\cos\lambda = \ddot{\lambda} \tag{2.16}$$

and

$$\ddot{\phi}\cos^2\lambda - 2\dot{\phi}\dot{\lambda}\sin\lambda\cos\lambda = 0, \tag{2.17}$$

which simplify to

$$\ddot{\lambda} = -\dot{\phi}^2\sin\lambda\cos\lambda \tag{2.18}$$

and

$$\ddot{\phi} = 2\dot{\lambda}\dot{\phi}\tan\lambda. \tag{2.19}$$

These differential equations cannot be solved so easily. However, some additional reasoning can show that a slider launched at an initial velocity on a frictionless sphere must travel in a great circle.

To see this, take $\dot{\lambda}$ and $\dot{\phi}$ to be constant velocities ω_λ and ω_ϕ. This is allowed because there are no forces acting on the slider after its initial launch, so there cannot be any acceleration. And since there is no acceleration, $\ddot{\lambda} = \ddot{\phi} = 0$. Then Eqs. 2.18 and 2.19 become

$$\omega_\phi^2\sin\lambda\cos\lambda = 0 \tag{2.20}$$

and

$$2\omega_\lambda\omega_\phi\tan\lambda = 0. \tag{2.21}$$

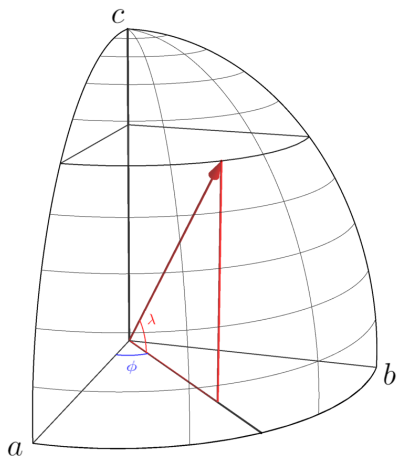


Figure 2.2: Schematic showing the coordinate system used to parametrize the equations of motion. The two relevant angles are λ, the latitude, and ϕ, the longitude.

The only value of λ that satisfies these equations is 0. But what if the slider starts at, say, $\lambda = \pi/4$? Since the sphere is symmetric, we can just redraw the coordinate system so that at this same point, λ is defined as zero. What this result really means is that the slider will move along a line of constant latitude, *if and only if the initial latitude is defined as* 0. This line is the equator, which is a great circle. Since this line could actually start anywhere on the sphere due to the symmetry of the sphere, this means that the slider will always move in a great circle.

2.2 Lagrangian of the Spheroid-Slider System

The equations of motion for a slider on a rotating spheroid are derived using the same method as above, but with considerably more assistance from *Mathematica*. The position of the slider on a triaxial ellipsoid is given in polar coordinates by

$$\vec{r} = \{a\cos\lambda\cos\phi, b\cos\lambda\sin\phi, c\sin\lambda\}, \tag{2.22}$$

where a, b, and c are the lengths of the three radii, λ is the latitude, and ϕ is the longitude, as shown in Fig. 2.2. If the two of the radii are equal, the ellipsoid is a spheroid. This project will cover the spheroid case only.

The vector $\vec{r}$ represents the position of the slider in the frame of the rotating spheroid (the non-inertial frame). To get the position in the inertial frame, the position vector is dotted with the rotation matrix associated with the axis of rotation, which is defined by the latitudinal angle λ. *Mathematica* can generate the appropriate matrix for rotation about any axis with its RotationMatrix feature, however, for the purposes of this project, we will primarily consider

rotation about the x-axis. In the case where the x-axis is the site of rotation,

$$R_x[\theta] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{2.23}$$

where θ is the angle through which the ellipsoid has rotated. Since $\theta = \omega t$,

$$\vec{r}' = R_x[\omega_x t]\vec{r} \tag{2.24}$$

and so

$$\vec{r}' = \begin{bmatrix} a\cos\lambda\cos\phi \\ b\cos\lambda\cos\omega t\sin\phi - c\sin\sin\omega t \\ c\cos\lambda\cos\omega t\sin\phi + c\sin\lambda\sin\omega t \end{bmatrix} \tag{2.25}$$

where λ, ϕ are functions of time.

Given an expression for the position of the slider, the Lagrangian can be found from the kinetic and potential energies

$$\mathcal{L} = T - U, \tag{2.26}$$

where T is the kinetic energy of the slider and U is its potential energy. T is again taken to be

$$T = \frac{1}{2}m\dot{\vec{r}}' \cdot \dot{\vec{r}}'. \tag{2.27}$$

The Lagrange is then expanded and the Euler-Lagrange equations 2.14 and 2.15 are found from *Mathematica's* EulerLagrange function. The equations are then solved for their double derivatives using Mathematica's Solve.

Gravity

In general, for a massive object, the potential V is given by an integral over volume

$$V \propto -\iiint \frac{d\mathcal{V}}{r}. \tag{2.28}$$

The gravitational potential energy of a spheroid is found using the method in Theory of the Potential [4]. For a sphere, the expression is

$$V_S = -\frac{4}{3}G\pi\rho R^2 \tag{2.29}$$

where G is the gravitational constant, R is the radius, and ρ is the density of the object. For the spheroid, this expression is significantly more complicated. In the prolate case

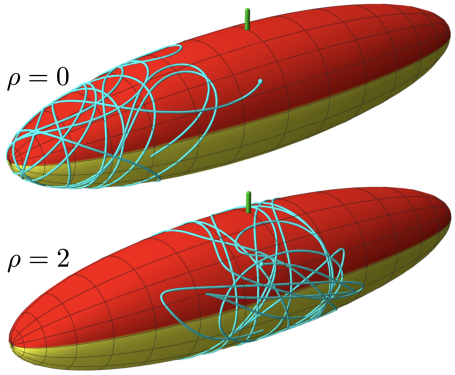


Figure 2.3: The effect of gravity on a slider on a highly eccentric ellipsoid. Gravity pulls the trajectory towards the center of mass.

$$V_P = -V_S \frac{\sqrt{1-r^2}(1-3\cos 2\lambda) + \sinh^{-1}\left(\sqrt{\frac{1}{r^2}-1}\right)\left(\left(r^2+2\right)\cos 2\lambda - 3r^2 + 2\right)}{2\left(1-r^2\right)^{3/2}} \tag{2.30}$$

where $r = \frac{a}{c}$. In this expression, the R in V_S is just r. The corresponding expression for the oblate case is

$$V_O = V_S \frac{\sqrt{r^2-1}(1-3\cos 2\lambda) + \sec^{-1}(r)\left(\left(r^2+2\right)\cos 2\lambda - 3r^2 + 2\right)}{2\left(r^2-1\right)^{3/2}} \tag{2.31}$$

These equations are defined separately to prevent the appearance of negative values under square roots. However, they can be treated as the same expression by *Mathematica*.

The gravitational potential has the effect of pulling the slider towards the center of mass. This is in opposition to the centrifugal pseudoforce, which drives the slider towards the poles

$$\vec{F}_{fict} = -m\vec{\omega} \times (\vec{\omega} \times \vec{r}). \tag{2.32}$$

The effect of the gravitational potential on a trajectory is illustrated in Fig. 2.3.

Meanwhile, the motion of the slider is constantly deflected perpendicular to its velocity by the Coriolis pseudoforce

$$\vec{F}_{Coriolis} = -2m\omega \times v. \tag{2.33}$$

The combined effect of the Coriolis and centrifugal pseudoforces can be seen in the simple case of a rotating sphere, with the slider starting at rest in the frame

of the sphere some distance away from the axis of rotation. The slider is pushed away from the axis of rotation and simultaneously deflected perpendicular to it, resulting in a trajectory that looks like a zipper (Fig. 2.4).

2.2.1 Viscosity

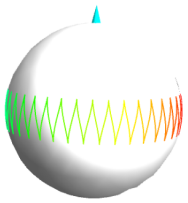


Figure 2.4: A slider starts from rest with respect to the body frame of a sphere and travels a zipper-like path. Red indicates early t.

Friction on the surface of the ellipsoid can be modeled by adding a viscosity term γ that is proportional to the speed of the slider. The deceleration at any given point in the trajectory due to the viscosity is then added to the second derivatives of longitude and latitude generated through the Lagrange method

$$\{\ddot{\lambda}_\gamma, \ddot{\phi}_\gamma\} = \{\ddot{\lambda} - \gamma\dot{\lambda}, \ddot{\phi} - \gamma\dot{\phi}\}. \tag{2.34}$$

2.3 Nondimensionalization

Nondimensionalization removes factors ρ and G, both of which have units. Nondimensionalization not only simplifies the equations but also makes it possible to simulate asteroids with known densities.

A natural dimensionless quantity t_N is defined as $t/\mathcal{T}$, where t denotes t units of time, and $\mathcal{T}$ is a quantity in terms of G and ρ with units of time

$$\mathcal{T} = \frac{1}{\sqrt{G\pi\rho}}. \tag{2.35}$$

(This time scale is proportional the period of a gravity train or surface skimming satellite on a planet with the given density, and is approximately 84 minutes for rock.)

The idea is to replace every appearance of t in the equations of motion with $t_N = t/\mathcal{T}$. Doing so cancels all dimensional quantities in the equations.

The following substitutions must also be made:

$$\Omega = \frac{\omega}{\sqrt{G\pi\rho}}, \tag{2.36}$$

$$\gamma_N = \frac{\gamma}{\sqrt{G\pi\rho}}, \tag{2.37}$$

$$v_{0_N} = \frac{v_0 c}{\sqrt{G\pi\rho}}. \tag{2.38}$$

These substitutions embed ρ and G in the values of Ω and γ, simplifying the calculation and reducing the number of parameters we have to consider.

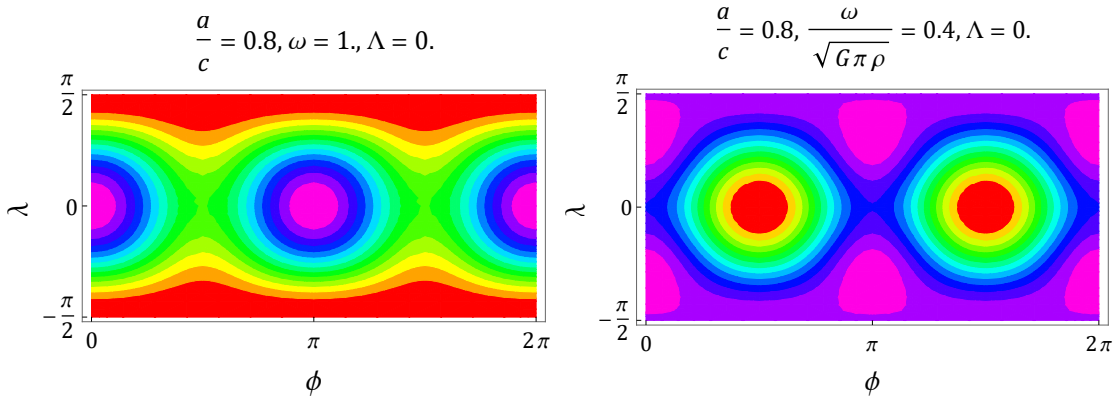


Figure 2.5: Contour plots comparing possible potential maps for the massless and massive cases. Left will not change with ω, but right will.

As strange as it sounds, this means that every possible potential surface on a massive rotating ellipsoid (this is not true for a massless ellipsoid; that case is separate) can be generated by choosing the correct value of Ω, because Ω in these nondimensionalized equations is not an angular velocity but actually a ratio with respect to the characteristic time $\mathcal{T}$.

The nondimensionalized equations are therefore describing the potential surface as a function of one ratio as opposed to calculating it from two independent values. So, every possible combination of ω and ρ is accounted for in these equations. Relatedly, all possible values of ω and ρ for which $\Omega = \frac{\omega}{\sqrt{G\pi\rho}}$ is the same will result in the same trajectory.

In the massless case, the relative potential surface on a given spheroid remains the same regardless of ω; as ω increases, the valleys become deeper and the hills become taller, but the distribution remains the same. In the massive case, however, multiple configurations of potential are possible (Fig. 2.5).

2.4 Initial Value Problem

In the case where the angle of the rotation axis $\Lambda = 0$, the nondimensionalized initial value problem is almost tractable. In the oblate case:

$$
\begin{aligned}
\lambda(0) &= \lambda_0, \\
\phi(0) &= \phi_0, \\
\dot{\lambda}(0) &= \frac{v_0 \cos\theta_0 \cos^2\lambda_0 \left(r^2 \tan^2\lambda_0 + 1\right)^{3/2}}{r\sqrt{r^4 \tan^2\lambda_0 + 1}}, \\
\dot{\phi}(0) &= v_0 \sin\theta_0 \sec\lambda_0 \sqrt{\frac{2}{(r^2-1)\cos 2\lambda_0 + r^2 + 1}}, \\
\ddot{\lambda}(t) &= -\gamma\dot{\lambda} + \frac{\sin 2\lambda}{2\left(r^2 \sin^2\lambda + \cos^2\lambda\right)} \left(\frac{-r^2\left(r^2+2\right)}{\left(1-r^2\right)^{3/2}} \operatorname{csch}^{-1}\left[\frac{r}{\sqrt{1-r^2}}\right] \left(3\cos^2\Omega t - 1 \right.\right. \\
&\quad + \frac{3r}{r^2+2}\left(r\cos 2\phi \sin^2\Omega t + 2\cot 2\lambda \sin\phi \sin 2\Omega t\right)\Big) + \frac{2r^4 - 7r^2 + 2}{2\left(r^2-1\right)} - \frac{1}{2}(r^2-2)\Omega^2 \\
&\quad + \left(1-r^2\right)\dot{\lambda}^2 - r\dot{\phi}\left(2\Omega\cot\lambda\cos\phi + r\dot{\phi}\right) + \frac{r^2}{2}\left(\frac{2r^2+1}{1-r^2} + \Omega^2\right)\cos 2\phi \\
&\quad \left. - \frac{2r^2+1}{r^2-1}\left(\left(r^2\sin^2\phi + 1\right)\cos 2\Omega t + 2r\cot 2\lambda \sin\phi \sin 2\Omega t\right)\right) \\
\ddot{\phi}(t) &= -\gamma\dot{\phi} - \frac{1}{r\left(1-r^2\right)^{3/2}} \left(-2\left(1-r^2\right)^{3/2}\dot{\lambda}\left(r\dot{\phi}\tan\lambda + \Omega\cos\phi\right) \right. \\
&\quad + 3r^2 \sinh^{-1}\left[\sqrt{\frac{1}{r^2}-1}\right]\left(r\sin 2\phi \sin^2\Omega t + \tan\lambda\cos\phi\sin 2\Omega t\right) \\
&\quad + \sqrt{1-r^2}\left(\cos\phi\Big(r\sin\phi\left(\left(r^2-1\right)\Omega^2 + \cos(2t\Omega) - 1\right) - \left(2r^2+1\right)\tan\lambda\sin 2t\Omega\Big)\right. \\
&\quad \left.\left. - 2r^3\sin 2\phi\sin^2 t\Omega\right)\right)
\end{aligned}
\tag{2.39}
$$

where v_0 is the initial speed in the spheroid frame and θ_0, an angle from the line of latitude, is the initial heading.

Trace

The Jacobian matrix

$$J = \frac{\partial f[\vec{x}, t]}{\partial \vec{x}} \tag{2.40}$$

contains all possible partial derivatives of the set of equations defined by

$$\dot{\vec{x}} = \vec{f}[\vec{x}, t]. \tag{2.41}$$

The Jacobian is a generalization of the partial derivative of a single function to a set of functions in multiple variables; as such, it describes the local behavior

of the functions. In the spheroid-slider system,

$$\vec{x} = \{\lambda, \phi, \dot{\lambda}, \dot{\phi}\} \tag{2.42}$$

$$\vec{f} = \{\dot{\lambda}, \dot{\phi}, \ddot{\lambda}, \ddot{\phi}\}. \tag{2.43}$$

The "trace" of a matrix refers to the sum of its diagonals. That makes the trace of the Jacobian of the spheroid-slider system

$$tr[J] = \frac{\partial \lambda}{\partial \dot{\lambda}} + \frac{\partial \phi}{\partial \dot{\phi}} + \frac{\partial \dot{\lambda}}{\partial \ddot{\lambda}} + \frac{\partial \dot{\phi}}{\partial \ddot{\phi}}. \tag{2.44}$$

If the Jacobian matrix of the initial value problem is evaluated and the trace taken, an expression independent of ϕ is obtained. The value of the expression is equivalent to the sum of the Lyapunov exponents associated with the system. We obtain the non-constant trace

$$\sigma = \frac{2\dot{\lambda} \tan\lambda \left(r^2 \tan^2\lambda - r^2 + 2\right)}{r^2 \tan^2\lambda + 1} - 2\gamma \tag{2.45}$$

for both the prolate and oblate cases.

Chapter 3

Chaos

3.1 Detecting Chaos

Chaos as a phenomenon can manifest in a variety of different ways, but it is always associated with sensitivity to initial conditions. In a chaotic system, slightly perturbing the initial conditions will give divergent results. In order for chaotic solutions to occur, the system of equations must be nonlinear and must have at least three degrees of freedom.

Chaos can be identified with qualitative and quantitative methods. The simplest is to look at a map of the trajectory and try to determine if the motion is repetitive. Chaotic motion does not tend to repeat itself. Such judgments can be misleading, however, if the trajectory looks like a big mess, but is actually just a small mess that repeats itself predictably. A more sophisticated, but still ultimately qualitative, method is to look at a Poincaré section of the motion.

3.1.1 Poincaré Sections

Poincaré sections represent the phase space of the motion with a two- or three-dimensional projection that is sampled according to some arbitrary strobing condition. Periodic motion appears on a Poincaré section as filled regions, continuous curves, or points, while chaotic motion looks "fuzzy" and/or has a fractal dimension. Fuzziness is indicative of a pattern of motion that is not repeating itself in an orderly or predictable way.

The driven pendulum is an example of a simple system with complicated Poincaré sections. It is a deceptively basic system in which chaos can be observed through careful manipulation of initial conditions. Applying a damping force gives rise to a strange attractor that displays fractal geometry in its phase space (Fig. 3.1). The chaotic behavior of the damped driven oscillator can be verified experimentally [6].

But Poincaré sections can be misleading if the strobing condition is not well-chosen. Typically, Poincaré sections of nonlinear motion are strobed on some periodic parameter of the system. For example, in the driven pendulum system,

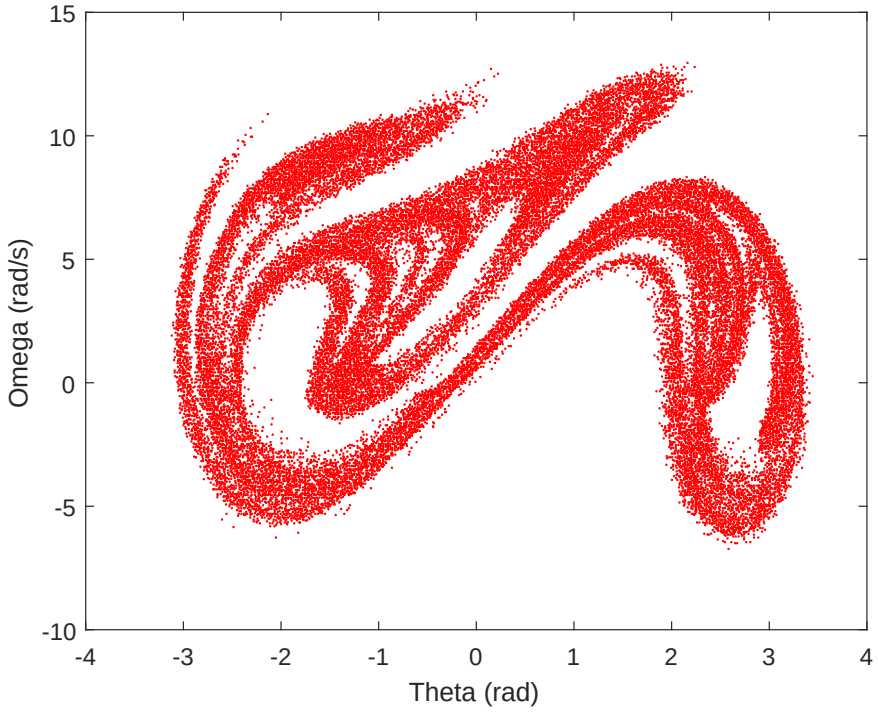


Figure 3.1: Poincaré section of a physical forced damped pendulum driven by a stepper motor at 0.785 Hz with a drive amplitude of 4.5 cm and a damping magnet 1.1 cm from the pendulum. The plot contains experimental data from 76400 drive cycles, as detailed in [6].

the most insightful Poincaré sections are obtained by strobing once per drive cycle. It would be natural to assume that the equivalent of the drive cycle for the spheroid-slider system would be the period of the rotation, but strobing on this condition results in sections that have a "fuzzy" look even for periodic orbits; they look like ordinary maps of the motion, but with points missing in between strobes (see the orange points in figure 3.2). I had to find a different strobing condition to get useful Poincaré sections.

Due to the geometry of the ellipsoid, trajectories on the curved surface diverge and re-converge as they travel along the curvature regardless of whether or not they are chaotic. The curvature also affects how quickly the trajectories separate; trajectories in regions of high curvature are driven apart more rapidly than trajectories in regions of low curvature. This behavior is reflected in the trace of the Jacobian matrix of partial derivatives associated with the system, which oscillates as a function of $\lambda(t)$ (see equation 4.4). Among the systems tested, this property appears to be unique to the ellipsoid-slider system.

Poincaré sections can take advantage of this oscillation by strobing latitude versus longitude every time the trace of the trajectorys Jacobian crosses the

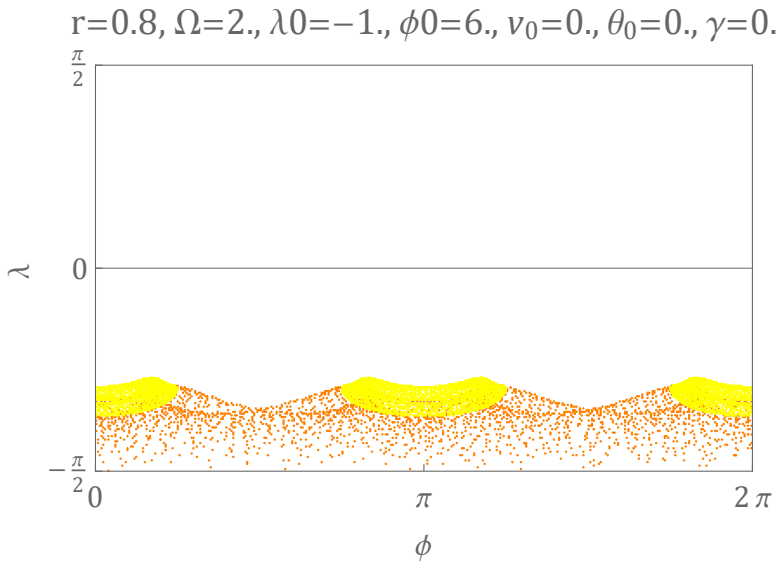


Figure 3.2: Poincaré section of a periodic orbit. The orange points were strobed on Ω while the yellow were strobed on the trace.

x-axis (equivalently, every time the Lyapunov exponents sum to zero). Strobing at the trace generates sections that reliably distinguish between chaotic and regular trajectories.

3.1.2 Lyapunov Exponent Calculation

Another way to confirm that an orbit is chaotic is to calculate its maximum Lyapunov exponent. This is easier said then done. The Lyaponuv exponents of a system represent the rate of divergence between infinitesimally close trajectories in each direction of the phase space. Thus, if the Lyapunov exponent is positive, the trajectories diverge and the system is chaotic. But if the exponent is negative, the trajectories converge to a steady state.

I use the Lyapunov exponent ל to check for chaos in our system. (The conventional notation is λ, but to avoid confusion with latitude, I use ל, or Lamed.) Chaotic trajectories yield positive Lyapunov exponents because their sensitivity to initial conditions is the result of an exponential divergence in $\delta\vec{x}$.

I use the method described in Parker and Chua [7] to derive the variational equation. We are given a set of differential equations

$$\dot{\vec{x}} = \frac{\partial \vec{x}}{\partial t} = \vec{f}[\vec{x}, t] \tag{3.1}$$

with initial condition

$$\vec{x}[t_0] = \vec{x}_0 \tag{3.2}$$

and we wish to know the rate at which solution to this initial value problem changes as a result of changes to the initial condition. Thus we take the partial derivative of both sides with respect to $\vec{x}_0$

$$\frac{\partial}{\partial \vec{x_0}}\Big(\frac{\partial \vec{x}}{\partial t} = f[\vec{x}, t]\Big) \tag{3.3}$$

and obtain

$$\frac{\partial}{\partial t}\Big(\frac{\partial \vec{x}}{\partial \vec{x_0}}\Big) = \frac{\partial f[\vec{x}, t]}{\partial \vec{x_0}}. \tag{3.4}$$

Applying the chain rule to the right hand side yields

$$\frac{\partial f[\vec{x}, t]}{\partial \vec{x_0}} = \frac{\partial f[\vec{x}, t]}{\partial \vec{x}} \frac{\partial \vec{x}}{\partial \vec{x_0}} \tag{3.5}$$

where we see that

$$\frac{\partial f[\vec{x}, t]}{\partial \vec{x}} = J, \tag{3.6}$$

the matrix of partial derivatives.

If we replace $\partial\vec{x}/\partial\vec{x}_0$ with Φ, which now represents the rate of change of $\vec{x}_0$ with respect to its initial condition,

$$\frac{\partial}{\partial t}\Big(\frac{\partial \vec{x}}{\partial \vec{x_0}}\Big) = \dot{\Phi} \tag{3.7}$$

and so

$$\dot{\Phi} = J\Phi, \tag{3.8}$$

which is the variational equation, a linear matrix equation whose initial condition is simply the identity matrix

$$\Phi[\vec{x}_0, t_0] = I. \tag{3.9}$$

Since $\Phi[\vec{x}, t]$ is a rate of change with respect to a perturbation $\partial\vec{x}_0$, multiplying the rate by the perturbation yields the total difference in $\vec{x}$ at time t as

$$\delta\vec{x} = \Phi\delta\vec{x}_0. \tag{3.10}$$

If the time derivative is taken, a set of ordinary differential equations is obtained

$$\delta\dot{\vec{x}} = \dot{\Phi}\delta\vec{x_0} = J\Phi\delta\vec{x_0} \tag{3.11}$$

which becomes, with the substitution $\Phi = \partial\vec{x}/\partial\vec{x}_0$

$$\delta\dot{\vec{x}} = J\delta\vec{x}. \tag{3.12}$$

The solution to the set of ordinary differential equations are exponentials of the form

$$\delta\vec{x} = e^{Jt}\delta\vec{x}_0. \tag{3.13}$$

The Lyapunov exponents are related to the eigenvalues of the Jacobian; the sum of the eigenvalues is equal to the sum of the Lyapunov exponents. By the eigenvalue-eigenvector property, $J\vec{\epsilon}_n = \lambda_n\vec{\epsilon}_n$, the independent eigenvector theorem, and the fact that $f(J)\vec{\epsilon}_n = f(\lambda_n)\vec{\epsilon}_n$,

$$e^{Jt}\delta\vec{x}_0 = e^{\lambda_1 t}c_1\vec{\epsilon}_1 + c_2e^{\lambda_2}\vec{\epsilon}_2 + ... + e^{\lambda_n t}c_n\vec{\epsilon}_1, \tag{3.14}$$

and so

$$e^{Jt}\delta\vec{x}_0 = c_1e^{\lambda_1 t}\vec{\epsilon}_1 + c_2e^{\lambda_2}\vec{\epsilon}_2 + ... + c_ne^{\lambda_n}\vec{\epsilon}_n \tag{3.15}$$

for an n-dimensional phase space. Each λ_n represents the rate of divergence in the nth dimension of the phase space, and each term gives the distance δx_n the adjacent trajectory has diverged from the fiducial trajectory in the nth dimension.

The sum of a set of exponential terms is dominated by the term with the largest exponent, so

$$\delta\vec{x} \approx \delta\vec{x}_0e^{\lambda t} \Rightarrow \lambda_{max} \approx \frac{1}{t}\log\left|\left|\frac{\delta\vec{x}}{\delta\vec{x_0}}\right|\right| \tag{3.16}$$

gives an approximation of the maximum Lyapunov exponent. While the spectrum of exponents can be enlightening, showing that $\lambda_{max} > 0$ is sufficient to argue that an orbit is chaotic.

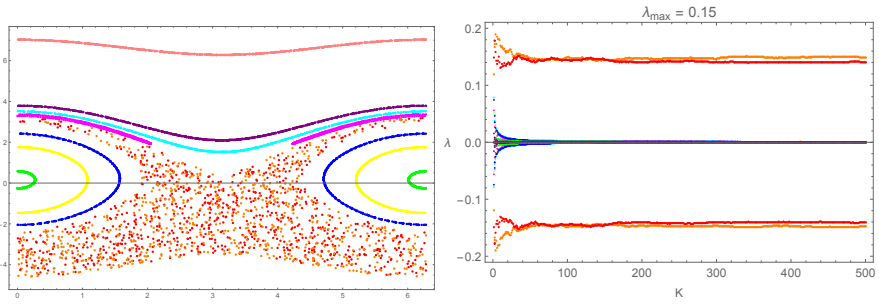


Figure 3.3: Left: a set of Poincaré sections of the undamped driven pendulum for various orbits. "Fuzzy" areas indicate chaos, while smooth lines indicate periodic orbits. Right: list plot showing a conventional Lyapunov spectrum calculation color-matched to each orbit in the Poincare section plot. Chaotic orbits have a positive exponent, while periodic orbits have exponents of zero. K is the number of renormalizations performed by the algorithm.

One of the hurdles encountered in the project was in applying the spectrum algorithm to the ellipsoid-slider system. The conventional application failed to

yield sensible results. In the process of refining the Lyapunov calculation, it was applied to a variety of simulated systems, including the pendulum (Fig. 3.3) and the Lorenz system (Fig. 3.5), and achieved good agreement with the literature (to within a decimal place, which is expected for Lyapunov calculations). While the algorithm demonstrated agreement with the literature when applied to these toy model systems, it failed on the ellipsoid-slider system.

To circumvent the problem, an alternative method was used in which a linear fit was applied to the log of the separations of two arbitrarily chosen trajectories a very small distance apart. The algorithm does not average over any renormalizations, instead plotting rate of increasing distance between the fiducial and adjacent trajectories. The slope of the fitted line approximates the maximum Lyapunov exponent. The algorithm agreed well enough with the conventional calculation for the purposes of detecting chaotic orbits.

This method reveals some interesting behavior; the slope is not always constant as expected, suggesting variable Lyapunov exponents that cannot be described by a constant value. (While it is normal for variation to occur at early times before the orbit stabilizes, in the case of the spheroid-slider system dramatic variation was seen late in the process.) Variable exponents could explain why the conventional algorithm failed on the system.

Results from the maximum Lyapunov exponent algorithm and the spectrum algorithm described in [7] are consistent for the undamped pendulum model system (Fig. 3.4) and the Lorenz system (Fig. 3.5).

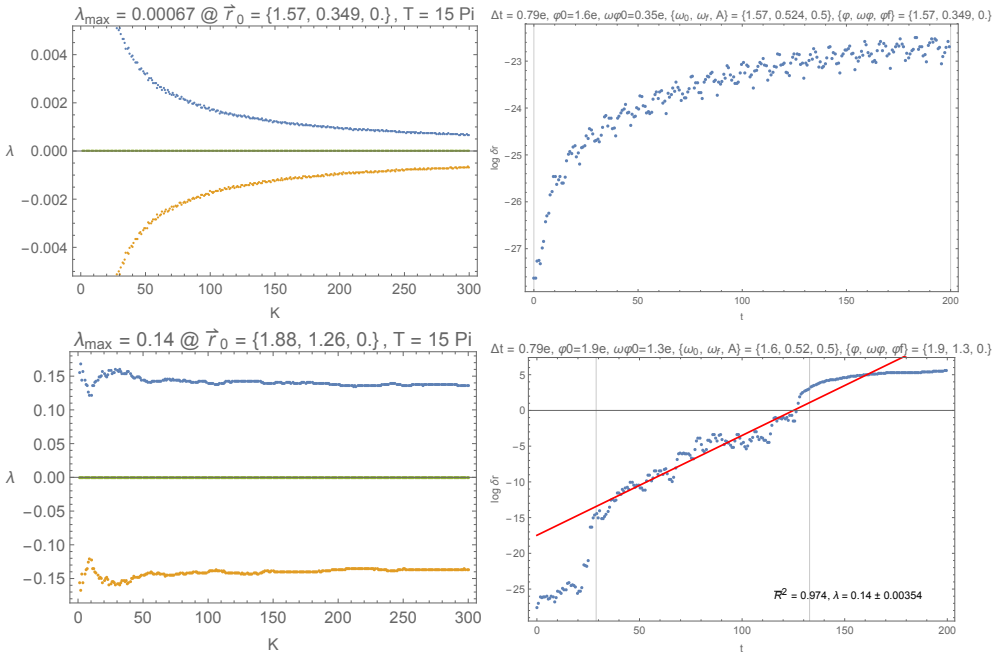


Figure 3.4: Comparison of methods used to calculate Lyapunov exponents for the undamped pendulum. Top: Comparison between the Lyapunov spectrum for a periodic pendulum orbit and the λ_{max} algorithm for the same orbit. Bottom: The same as above for a chaotic orbit. The spectrum algorithm gives $\lambda_{max} \approx 0.14$ and the λ_{max} algorithm gives $\lambda_{max} \approx 0.190 \pm 0.003$ for the pendulum.

3.2 The Maximum Lyapunov Exponent of the Lorenz System

Before applying the algorithm to the spheroid-slider system, I tested it on the Lorenz system, the Lyapunov spectrum of which is well-documented. Other studies have shown that λ_{max} of the Lorenz system is around 0.9, depending on the numerical methods used [8]. My algorithm estimated $\lambda_{max} = 1.00 \pm 0.02$, a reasonable approximation considering that this algorithm does not renormalize at any points along the trajectory.

For constant λ, plotting the log of the final distance between points along two trajectories over the initial distance between the trajectories as a function of t should yield a linear graph, the slope of which approximately equals λ_{max}.

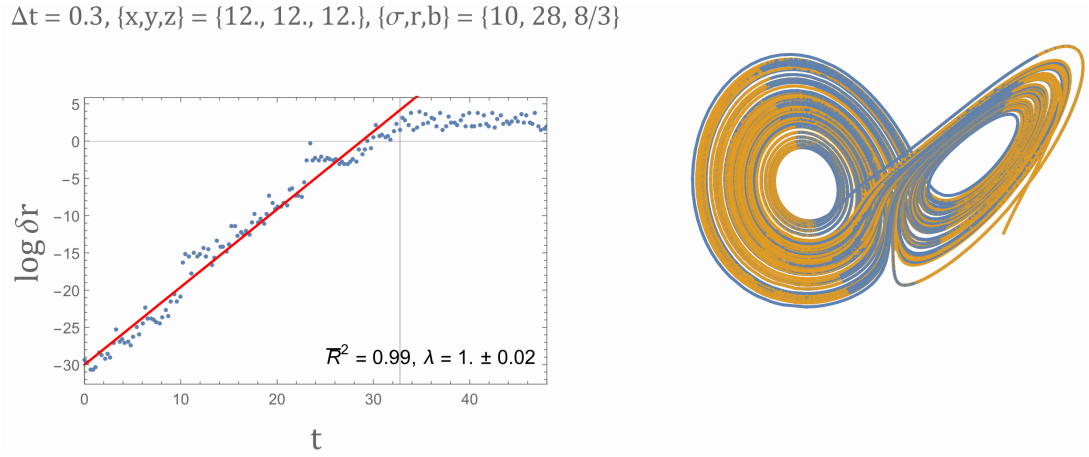


Figure 3.5: The maximum Lyapunov exponent $\lambda = 1.00 \pm 0.02$ calculated for the Lorenz system with initial conditions {x,y,z}={12,12,12}. Both fiducial and adjacent trajectories are plotted on the right. In this case, they are too close together to see the difference clearly.

3.3 Fractals

In general, "fractal" refers to a shape that exhibits self-similar geometry at increasing magnification. A very simple fractal is the Koch snowflake. The spiral shape of the Lorenz system is also a fractal.

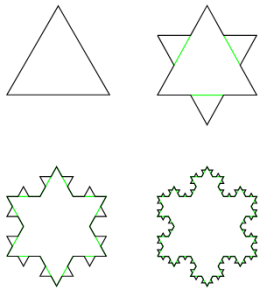


Figure 3.6: The first four iterations of the Koch snowflake. CC BY-SA 3.0, commons.wikimedia.org

Fractal geometry is associated with dissipative nonlinear systems. Fractal patterns arise when systems are drawn to what are called "strange attractors." In a physical system, the attractor is the basic pattern of motion or dynamical evolution that the system tends to conform to. The properties of an attractor arise from the forces associated with the system. In a deterministic system, the "attractor" is a well-defined point or curve. The attractor of the Earth's orbit around the sun is an ellipse, while the attractor of a ball rolling on a surface with friction is a point, because the friction eventually damps out the motion entirely.

Such attractors have integral dimensions. But in some systems, an attractor with fractal dimensions–a strange attractor–is possible. A system converging on a strange attractor will usually exhibit characteristically chaotic behavior.

In the case of an ellipsoid of revolution I could not find conclusive evidence of strange attractors when dissipative friction is added to the system; the trajectories tend to converge on points associated with regions of low potential energy, or basins. However, this does not mean strange attractors do not exist in the system; they may simply be rare or difficult to identify.

Fractal basin boundary plots can show sensitive dependence on initial conditions and hint at deeper structures. When dissipative friction is added to our system, trajectories may converge on a point attractor. So far, attractors have been seen only at the poles. When the outcome of a trajectory starting at a certain point is determined for every point on the spheroid, a binary map can be created to represent where trajectories will eventually land given their starting point. It turns out that fractal patterns emerge in these maps, which are discussed further in the results section.

3.4 Numerical Integration and Accuracy of Results

Mathematica's NDSolve function was used to compute orbits from the initial value problem. While the working precision is sufficient to accurately compute orbits in early t, inaccuracy does occur if higher precision settings are not used. The inaccuracies are imperceptible in periodic orbits, but for chaotic orbits,

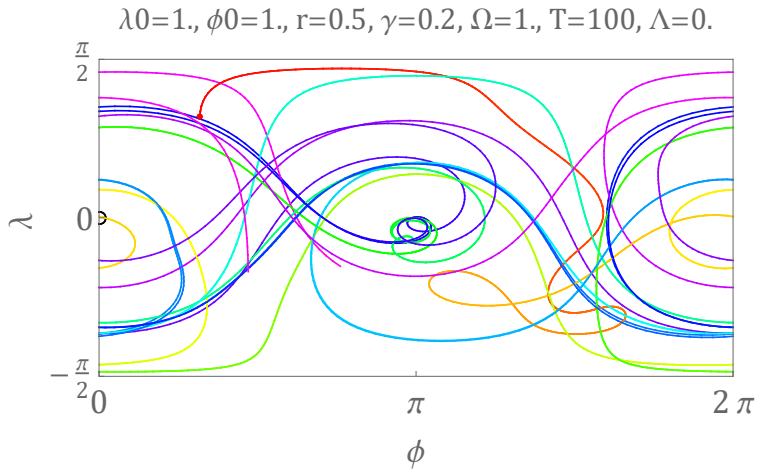


Figure 3.7: A chaotic trajectory run until $t = 100$. Trajectories are color-coded according to time, with red corresponding to $t = 0$. The trajectory computed at machine precision is layered over the trajectory computed at 3 times machine precision. The difference becomes perceptible in the blue phase and becomes more dramatic as the trajectory progresses to violet.

which are highly sensitive to small deviations, dramatic inaccuracy can occur, as seen in Fig. 3.7.

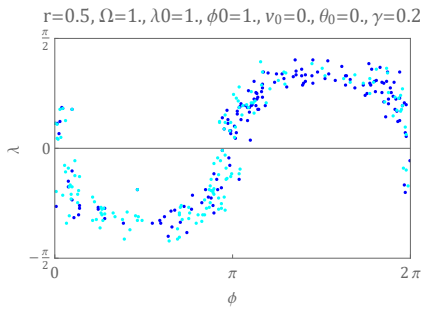


Figure 3.8: Poincaré section of the $r = 1/2, \Omega = 1$ spheroid computed with 3 times machine precision (cyan) layered over the same computation at machine precision (dark blue) for a run time of $t = 10^3$. While the points are in slightly different positions, the pattern is the same. These sections wre strobed on the trace.

While the trajectories may look superficially different, the patterns seen in associated Poincaré sections and fractal basin plots are consistent even for less precise runs (provided that the Poincaré sections do not run for too long) (Fig. 3.8). This lends credibility to the results, as it is evidence of consistency in the calculations.

Chapter 4

Results

4.1 Variable Lyapunov Exponents

We expected to see linear, uninterrupted scaling regimes when using the method described in the previous chapter. However, this assumes the rate of divergence described by the Lyapunov exponent will remain constant along a trajectory. It turns out this is not the case for our particular system, but the results do show that local trajectories diverge exponentially, which is sufficient to prove that there are chaotic orbits on the spheroid.

4.1.1 Relationship to the Trace of the Jacobian

The Jacobian matrix is not related to the Lyapunov exponents directly. However, the sum of the eigenvalues of the Jacobian, or the trace, is equivalent to the sum of the Lyapunov exponents. Recall that in the spheroid case, the curvature of the ellipsoid is constant about ϕ and varies about λ. That the sum of the Lyapunov exponents σ (Eqn. 4.4) depends on the latitude suggests that the changing curvature plays a significant role in the manifestation of chaotic orbits.

For a system of constant Lyapunov exponents, the sum of those exponents should be positive. However, here the Lyapunov exponents are clearly changing in relation to λ, the direction of changing curvature on an ellipsoid of revolution. As a result, the calculation of the maximum exponent is not necessarily reliable.

4.1.2 Results from the Graphical Lyapunov Exponent Algorithm

For constant $\mathfrak{h}$, plotting the log of the final distance between points along two trajectories over the initial distance between the trajectories as a function of t should yield a linear graph, the slope of which approximately equals $\mathfrak{h}_{max}$. In many cases, the distance either does not grow significantly or the plot scales

linearly with time before reaching a plateau at which the maximum distance between trajectories has been reached (Fig. 4.1).

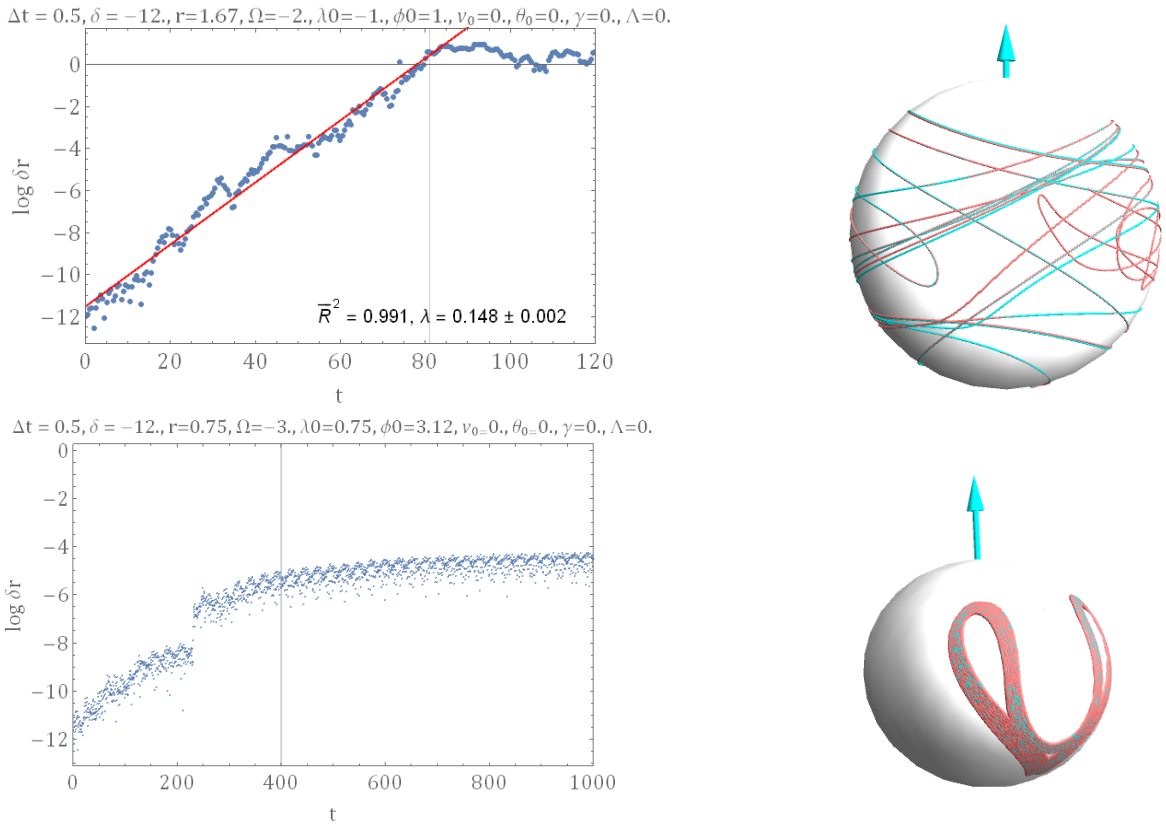


Figure 4.1: Top: maximum Lyapunov exponent $\lambda = 0.148 \pm 0.002$ calculated for a slider released from rest at an initial latitude of 1 and initial longitude of 2 on an oblate spheroid with aspect ratio $r = 5/3$ rotating at $\omega = -1.5$. The distance between the fiducial and adjacent trajectories (shown to the right in cyan and pink), initially 10^{-12} radians apart in latitude, was sampled every $\Delta t = 0.5$ units of time. There is a linear scaling regime associated with these parameters. The plateau around $t = 80$ results from the bounded nature of the spheroid and represents the maximum possible separation between trajectories. Bottom: Periodic orbit of a a slider released from rest at an initial latitude of 0.75 and initial longitude of 3.12 on a prolate spheroid with aspect ratio $r = 3/4$ rotating at $\omega = -3$. The trajectories do not separate significantly, even after $t = 1000$.

However, in a few cases, the regime does not scale linearly; there may be more than one distinct linear regime, or parts of the graph that are curved. In these cases it is not possible to characterize the trajectory using only one

Lyapunov exponent. Figure 4.2 is a dramatic example of this result; the orbit appears to be periodic until around $t = 300$, when the distance δr between the trajectories suddenly starts to grow.

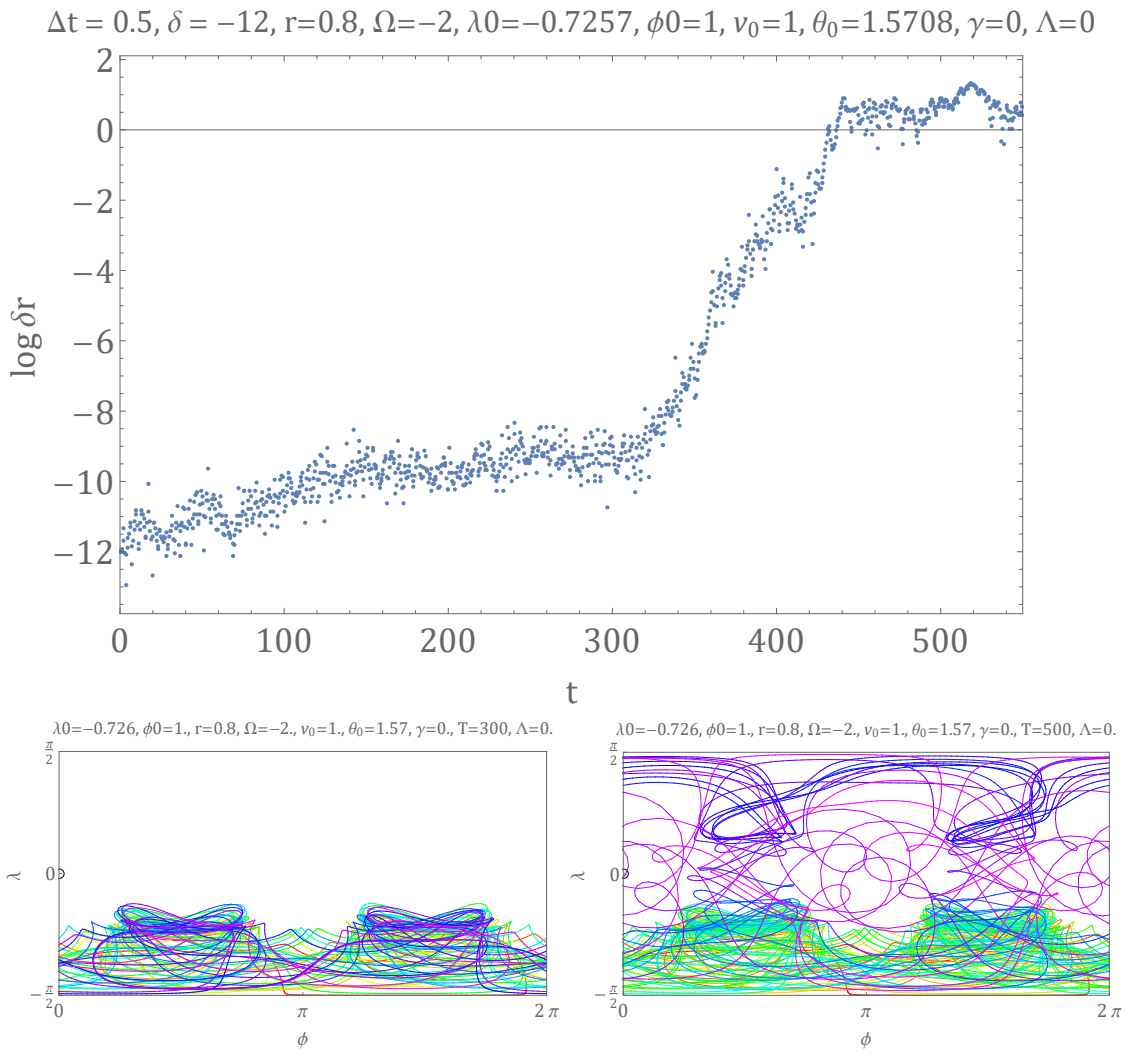


Figure 4.2: These orbits were calculated with a working precision of 3 times the machine precision. Nonlinear scaling regime of the orbit of a slider released from rest at an initial latitude of -0.7527 and initial longitude of 1 on a prolate spheroid with aspect ratio $r = 4/5$ rotating at $\omega = -2$. The distance between the fiducial and adjacent trajectories was sampled every $\Delta t = 0.5$ units of time. Bottom: Two dimensional projections after $t = 300$ (left) and $t = 500$ (right) of the fiducial orbit colored according to time, starting with red at $t = 0$ and ending with violet. The changes in the patterns of the motion reflect the changes in the scaling regime.

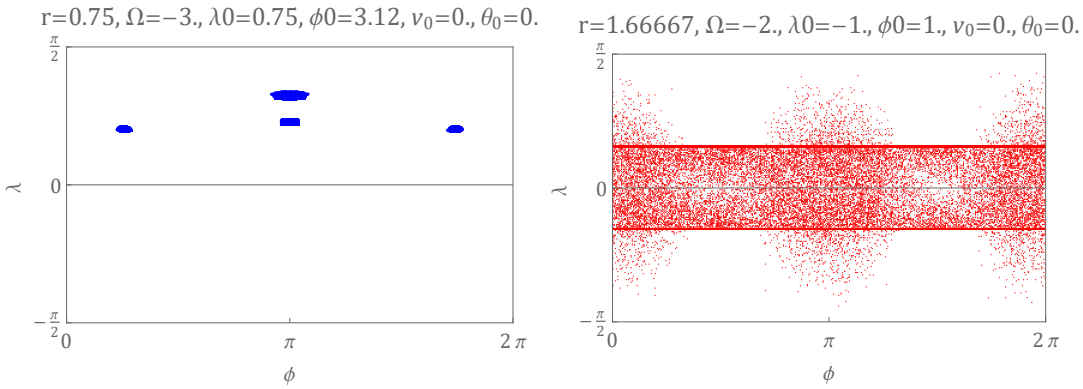


Figure 4.3: Poincaré sections of a periodic (left) and chaotic (right) trajectory. These are the same two spheroids in Figure 4.1.

4.2 Poincaré Sections

The Poincaré sections obtained for the spheroid-slider system have distinctly different features for periodic versus chaotic orbits (Fig. 4.3), but only when strobed on the zero of the trace (Fig. 4.4). I have yet to find a fractal structure to these sections, even when viscosity is nonzero.

4.2.1 Poincaré Sections With Viscosity

Curiously, chaotic orbits that occur with viscosity (that is, those orbits which do not spiral into point attractors) often have the same shape for the same viscosity, both when strobed on the trace and when strobed on the revolution (Fig. 4.5). These orbits only occur on sufficiently eccentric spheroids; orbits on mildly eccentric spheroids will always spiral to points.

4.3 Conditions for Chaos versus Periodicity

From qualitative reasoning and observations, it is clear that the outcome of a trajectory is dependent on two fundamental factors: curvature and potential energy. Chaos is observed neither on the sphere nor the static ellipsoid.

The hills and valleys of potential that arise from rotating reference frame perturb the slider from its geodesic path. As expected, trajectories that have their starting points near a region of low potential energy are more likely to settle into a periodic orbit around that well than enter a chaotic orbit. Chaotic orbits tend to be global rather than local; covering a large section of the spheroid. However, strikingly orderly global orbits can still be observed.

Chaos is also never seen on ellipsoids rotating about the axis of symmetry.

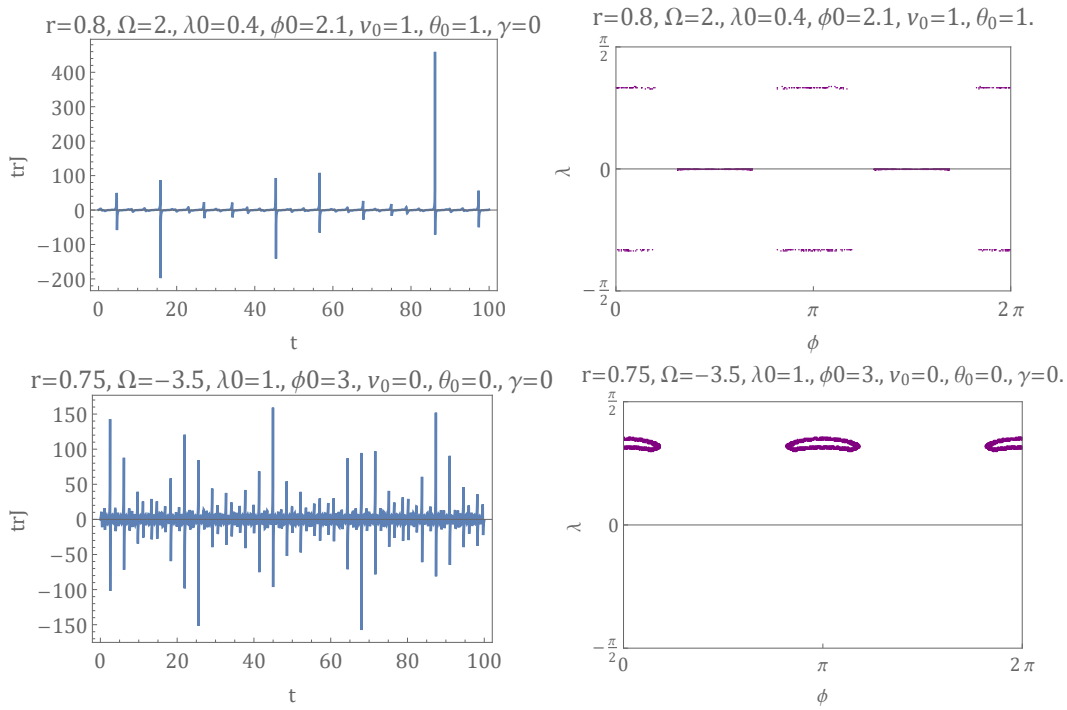


Figure 4.4: Top: The trace of the Jacobian for the orbit with initial conditions $\{\lambda_0, \phi_0, v_0, \theta_0\} = \{0.4, 2.1, 1, 1\}$ and $\{r, \Omega\} = \{4/5, 2\}$ (left) with the associated Poincaré section (right). Bottom: The trace of the Jacobian for the orbit with initial conditions $\{\lambda_0, \phi_0, v_0, \theta_0\} = \{1, 3, 0, 0\}$ and $\{r, \Omega\} = \{3/4, -3.5\}$ (left) with the associated Poincaré section (right).

4.4 Fractal Basins

When a damping term is added the system in the form of viscosity, attractors may emerge at two of the four poles defined by $\lambda = \pm\pi/2$ or $\phi = 0, 2\pi$. The attractors correspond to regions of low potential. For parameters where the centrifugal potential is less influential than the gravitational potential, the attractors are $\phi = 0, 2\pi$, and for parameters that emphasize the centrifugal potential (such as rapid spin), the attractors are seen at $\lambda = \pm\pi/2$.

This means that any trajectory, regardless of initial coordinates, will eventually spiral to a halt near one of the poles. In figure 4.8, these poles are shown in red. We designate the two possible outcomes with a 1 or a 0 and plot the outcomes of trajectories all over the spheroid to get fractal basins. The trajectories run for a set time, referred to here as the cycle time, before the attractor is determined.

We calculated fractal basins plots for both the massive and massless spheroid. The massless case is straightforward: orbits will always spiral to one point

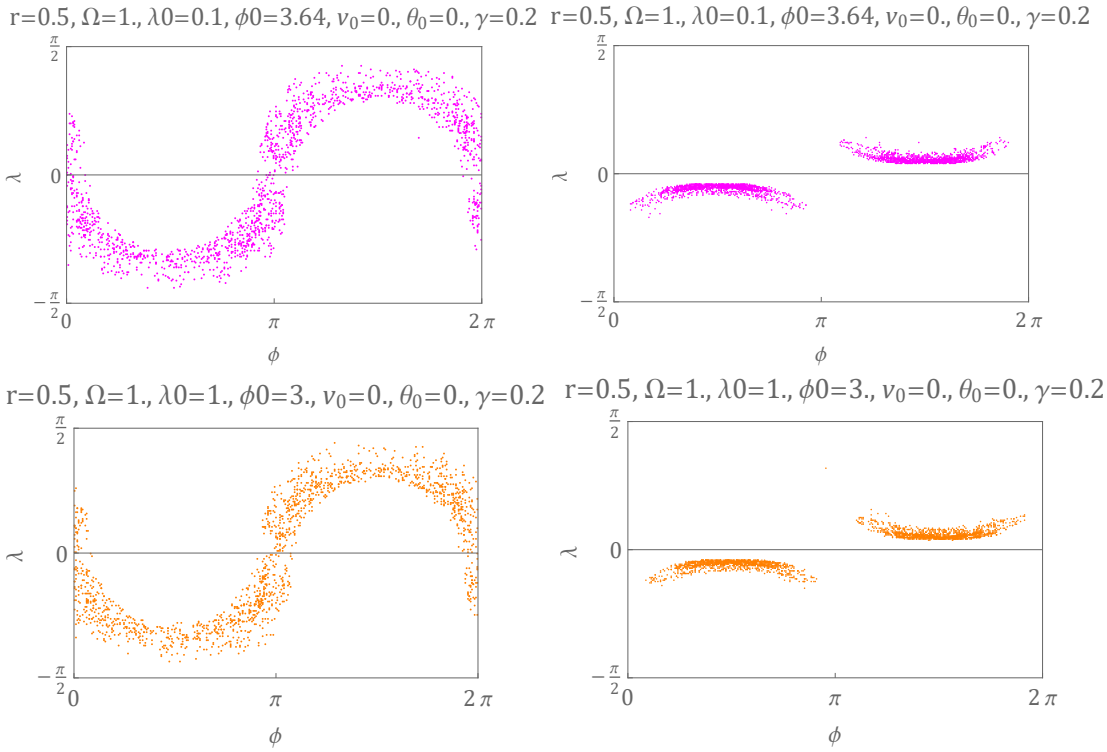


Figure 4.5: Top: side by side comparison of Poincaré sections strobed on the revolution (left) and on the trace (right) for the same initial conditions $\{\lambda_0, \phi_0, v_0, \theta_0\} = \{0.1, 3.64, 0, 0\}$ on a viscous spheroid with $\gamma = 0.2$, $r = 1/2$, and $\Omega = 1$. Bottom: Different initial conditions $\{\lambda_0, \phi_0, v_0, \theta_0\} = \{1, 3, 0, 0\}$ produce the same sections as above.

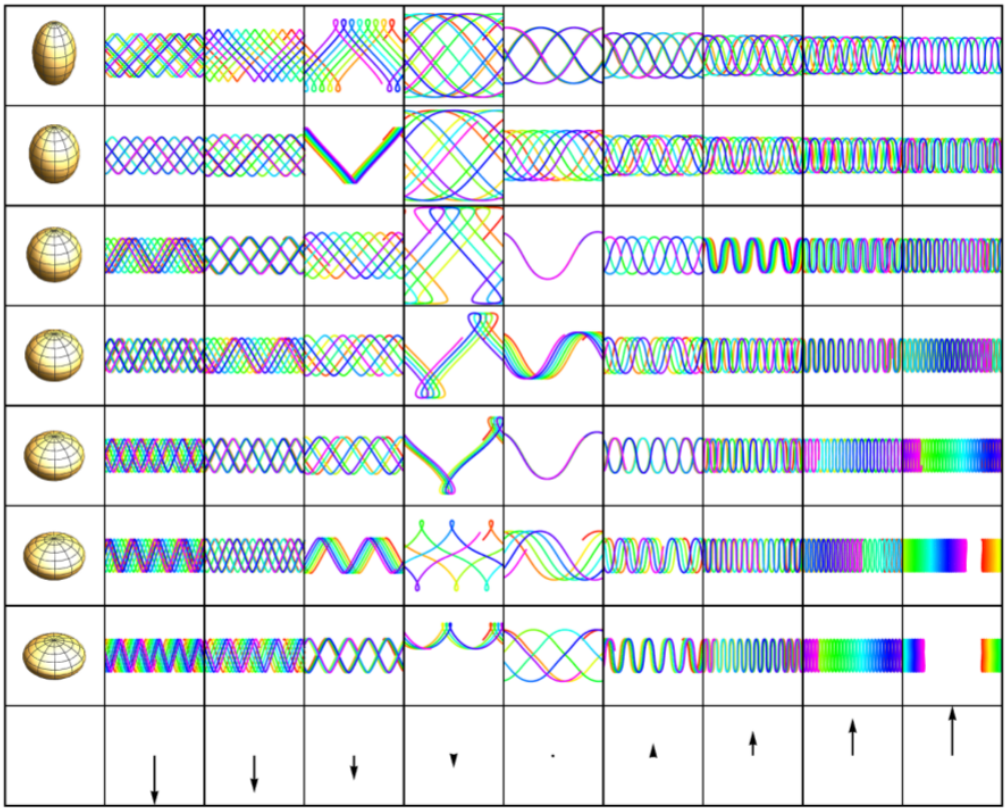


Figure 4.6: Grid showing the orbits for a variety of Ω around the axis of symmetry, where $\Lambda = \pi/2$. All of these orbits are clearly periodic. Note that they are not symmetric about $\Omega = 0$ because they have an arbitrary initial velocity in the frame of the spheroid.

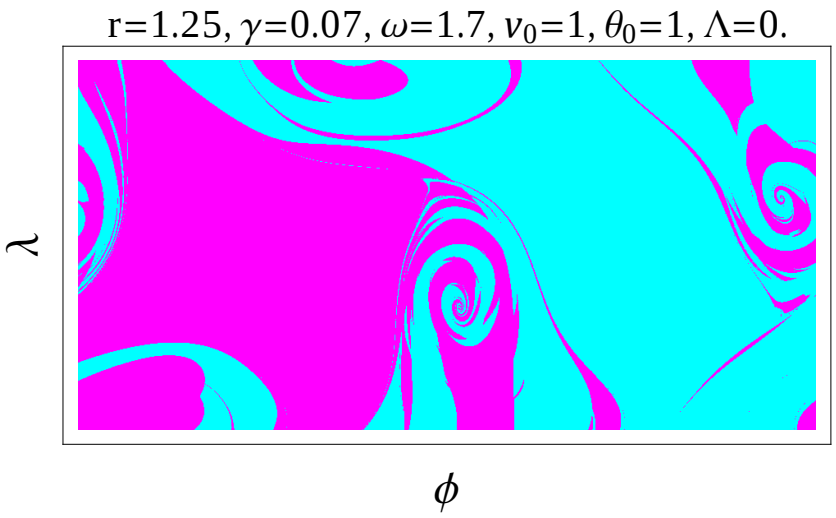


Figure 4.7: Fractal basin plots for a massless oblate spheroid with aspect ratio $r = 1.25$, $\Omega = 1.7$, and viscosity $\gamma = 0.07$.

attractor. This is not always the case for massive spheroids, which may have higher dimensional attractors (Fig. 4.5), so fractal basin plots are only an approximation of the true behavior. Fractal basins of massive spheroids tend to be more complex (compare the left and right images in Figure 4.11).

In general, the gravitational potential of spheroids of lower eccentricity has less of an effect on the motion of the slider than the same for highly eccentric spheroids, because the gradient of the potential is more extreme.

Decreasing the viscosity increases the complexity of the fractal, as seen in Figure 4.9.

For $\lambda = 0$, sliders released from rest always generate simple spiral basins. However, changing the launch speed (Fig. 4.10) and varying the angle (Fig. 4.12) results in distortions of the spiral form.

4.5 Regions of Chaos

I can crudely estimate which initial coordinates on the spheroid will generate chaotic trajectories by using the machinery of the maximum Lyapunov exponent algorithm. While it is unfortunately not practical to attempt to estimate λ_{max} from initial conditions at every point on the sphere, we know that if the fiducial and adjacent trajectories achieve a separation on the order of 1, it is safe to assume the orbit is chaotic.

The problem is that it may take several thousand T for more subtly chaotic orbits to reach this point (to say nothing of the numerical error that is bound to accumulate by this time), so we arbitrarily define as chaotic any orbit that

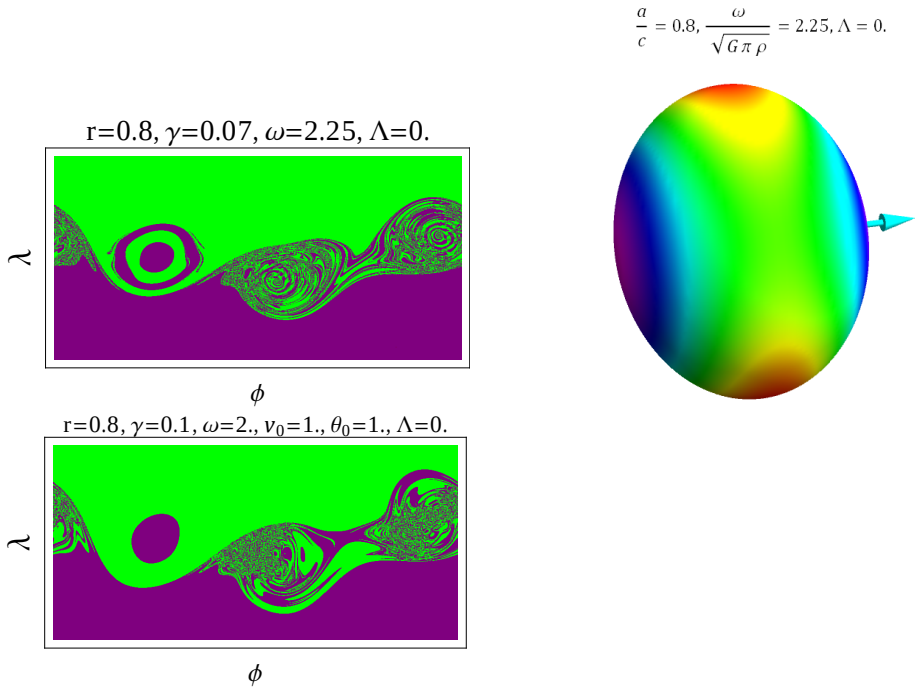


Figure 4.8: Left: Fractal basin plot for a massless (top) and massive (bottom) prolate spheroid with aspect ratio $r = 4/5$. The slider has been launched with initial velocity $v_0 = 1$ at angle $\theta_0 = 1$. Right: The potential surface on the spheroid. The attractors corresponds to the regions of lowest potential, seen here in red.

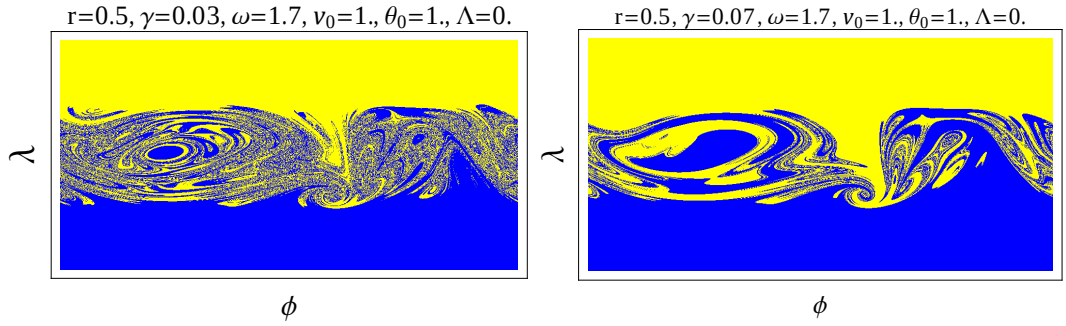


Figure 4.9: Comparison between lower and higher viscosities with a massless spheroid. Bottom: Fractal basin plots for a prolate spheroid with aspect ratio $r = 0.5$, $\omega = 1.7$, and viscosity $\gamma = 0.03$ (left) and $\gamma = 0.07$ (right). The slider has been launched with initial velocity $v_0 = 1$ at angle $\theta_0 = 1$. The resolution is $\delta = \pi/450$, and the cycle time is $t = 40$.

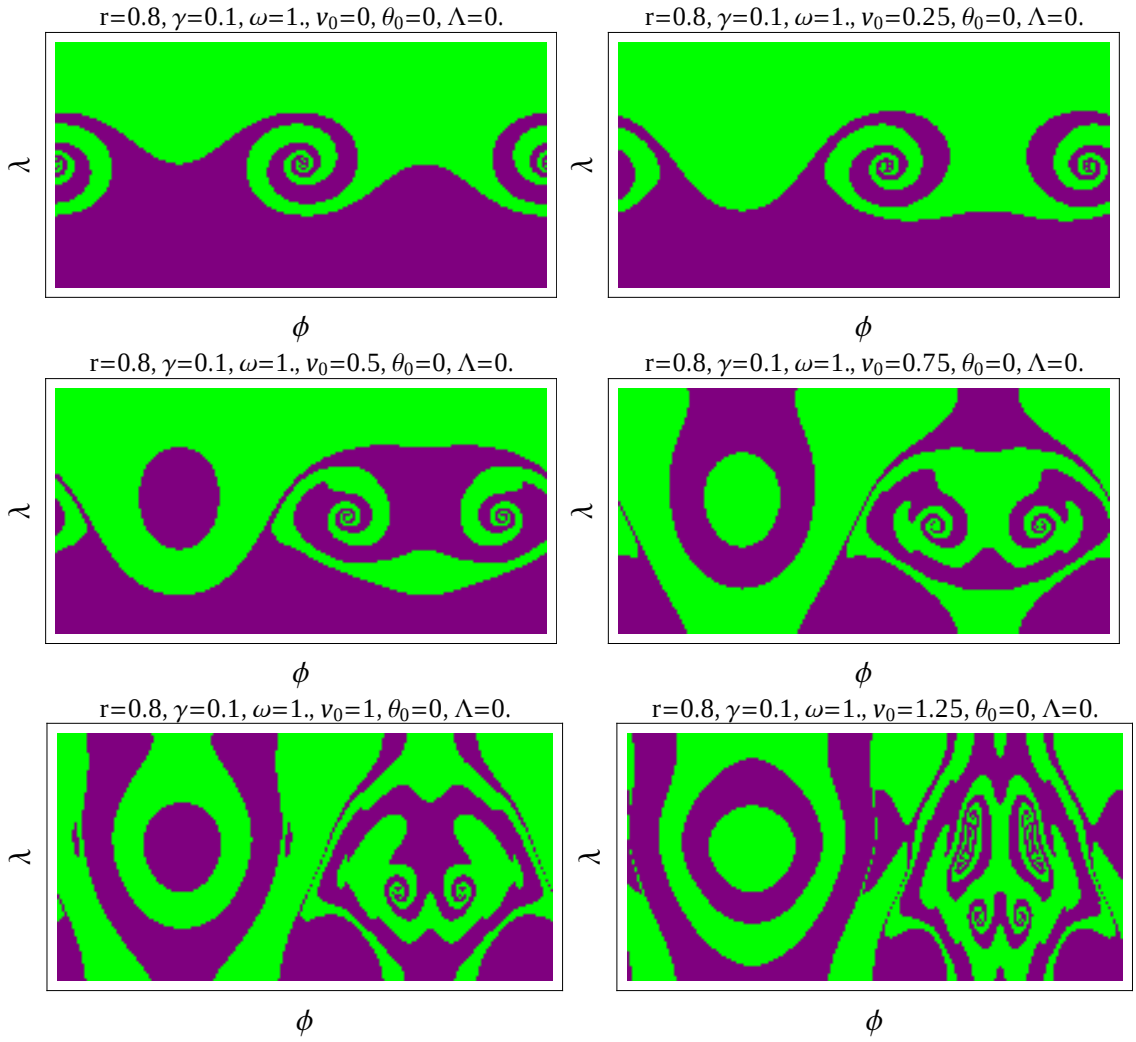


Figure 4.10: Effect of the initial velocity. Fractal basin plots for a prolate spheroid with $r = 0.8$, $\omega = 1$, and $\gamma = 0.1$. The slider has been launched at angles $\theta_0 = 0$ with increasing v_0. The resolution is $\delta = \pi/100$, and the cycle time is $t = 40$. The attractors are the latitudinal poles.

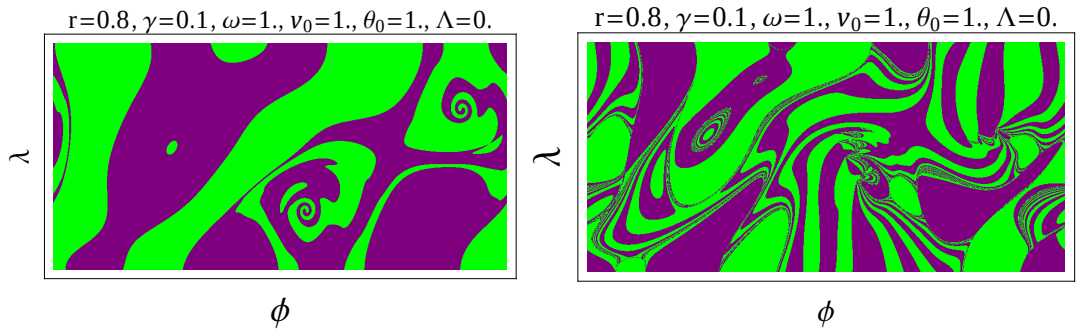


Figure 4.11: Comparison of the fractal basins for a massless (left) and massive (right) prolate spheroid with $r = 4/5$, $\Omega = 1$, and $\gamma = 0.1$. The slider has been launched at angles $\theta_0 = 0$ with increasing v_0. The resolutions are $\delta = \pi/450$ (left) and $\delta = \pi/400$ (right), and the cycle times are $t = 40$ (left) and $t = 100$ (right). The attractors are the latitudinal poles.

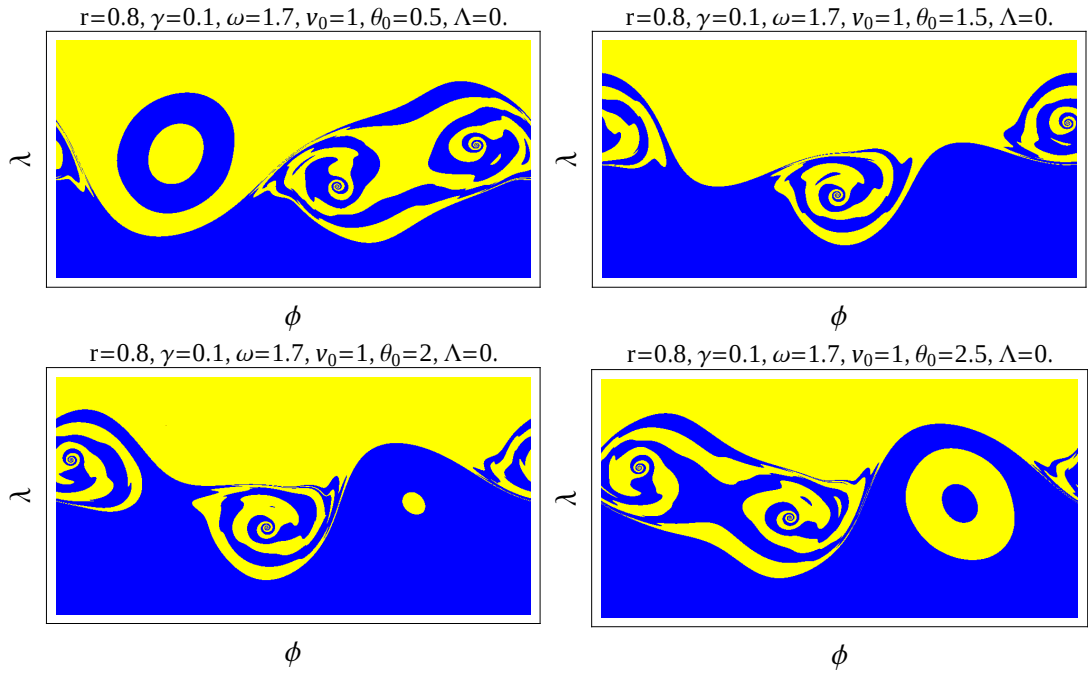


Figure 4.12: Effect of the initial launch angle. Fractal basin plots for a prolate spheroid with aspect ratio $r = 0.8$, $\Omega = 1.7$, and viscosity $\gamma = 0.1$ The slider has been launched with initial velocity $v_0 = 1$ at angles $\theta_0 = 0.5$, $\theta_0 = 1.5$, $\theta_0 = 2$, and $\theta_0 = 2.5$. The resolution is $\delta = \pi/450$, and the cycle time is $t = 40$. The attractors are the latitudinal poles.

achieves a separation of 10^x or greater within a time τ. x and τ can be determined on a case by case basis.

Manual inspection with the maximum Lyapunov algorithm shows that most of the time, if the fiducial and adjacent trajectories are at a significant distance δr by a reasonably early time t, that trajectory is likely to be chaotic. This method of distinguishing periodic from chaotic orbits benefits from fine tuning on a case-by-case basis, but in general, requiring $\delta r = 10^{-1/2}$ by $t = 600$ is a suitable metric for reasonable values of r and Ω.

When the initial coordinates of the trajectories for which this is true on a given ellipsoid are plotted, patterns emerge that echo the fractal basins shown in the previous section. These similarities can be seen in Figure 4.13 and 4.14. I will refer to such plots as "divergence plots" because they reflect whether or not the fiducial and trajectories have diverged. In order to avoid spurious results owing to numerical pathology near the poles, the plots range of -1.5 to 1.5 in the latitude.

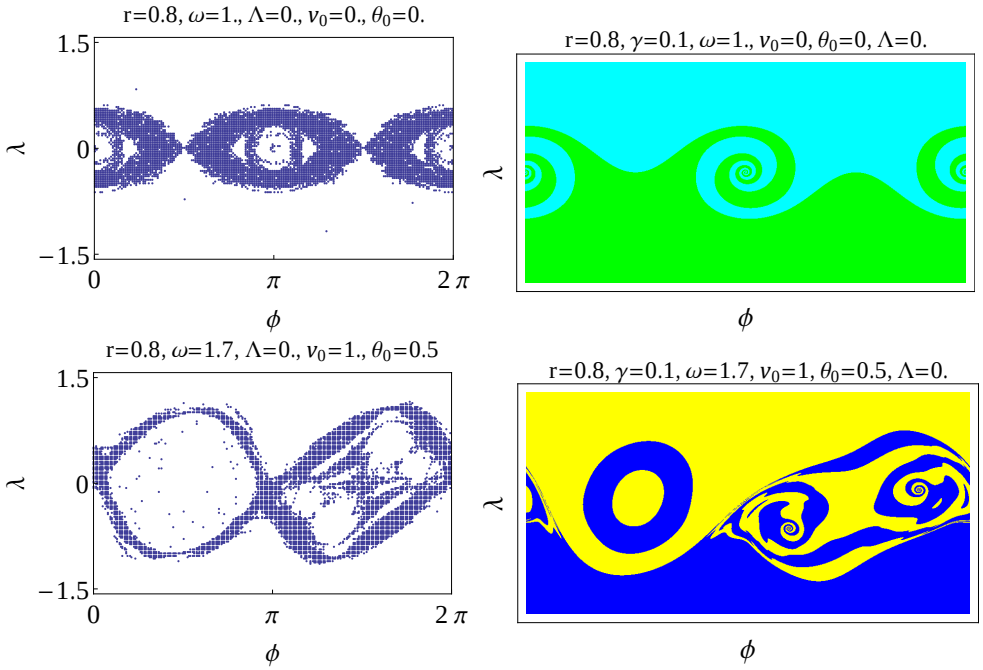


Figure 4.13: Top: Massless prolate spheroid of aspect ratio $r = 4/5$ and $\Omega = 1$. Points are all initial coordinates for which the fiducial and adjacent trajectories, after an initial separation of 10^{-12} radians, were at least $10^{-1/2}$ radians distant from one another by $t = 600$. The resolution is $\delta = \pi/500$. The associated fractal basin plot is shown on the right for comparison. Bottom: massless spheroid of aspect ratio $r = 4/5$ and $\Omega = 1.7$, with the slider launched at initial velocity $v_0 = 1$ at an angle $\theta_0 = 0.5$. All initial coordinates for which the fiducial and adjacent trajectories, after an initial separation of 10^{-12} radians, were at least $10^{-1/2}$ radians distant from one another by $t = 600$. The resolution is $\delta = \pi/100$. The associated fractal basin plot is shown on the right for comparison.

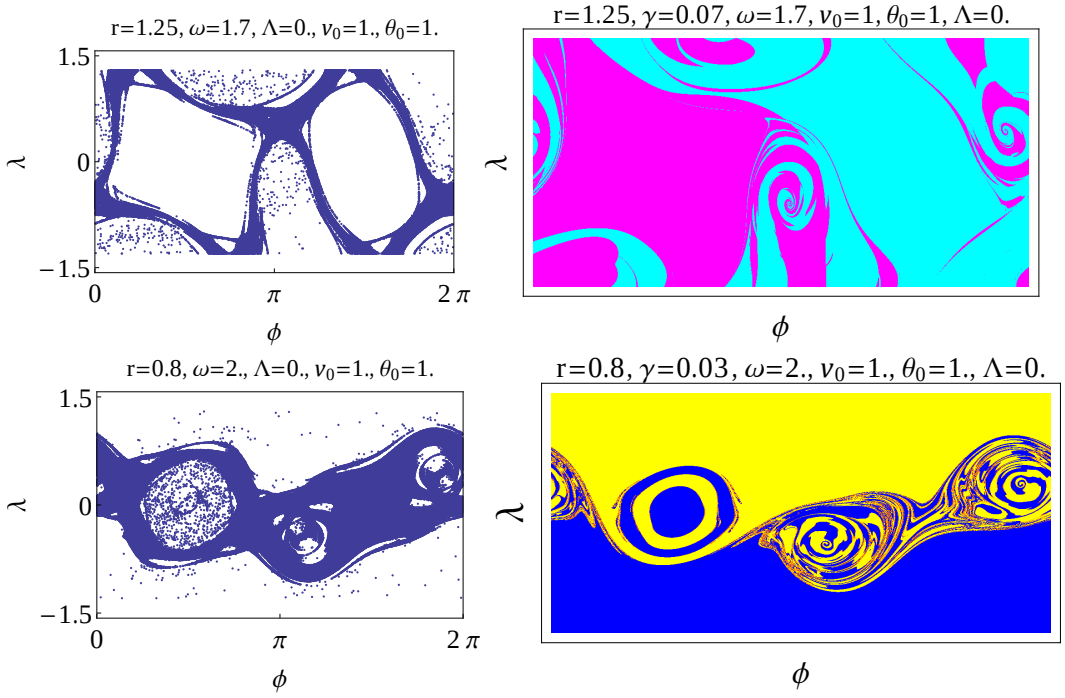


Figure 4.14: Top left: massless spheroid of aspect ratio $r = 5/4$ and $\omega = 1.7$. All initial coordinates for which the fiducial and adjacent trajectories, after an initial separation of 10^{-12} radians, were at least $10^{-1/2}$ radians distant from one another by $t = 600$. The resolution is $\delta = \pi/500$. Top right: Associated fractal basin plot for this spheroid, with $\gamma = 0.07$ and resolution $\delta = \pi/450$. Bottom left: divergence plot of a massless spheroid of aspect ratio $r = 4/5$ and $\Omega = 2$, with the slider launched at initial velocity $v_0 = 1$ at an angle $\theta_0 = 1$. Bottom left: associated fractal basin plot for the spheroid, with $\gamma = 0.03$ and resolution $\delta = \pi/600$.

Divergence plots of the massive spheroid show that the introduction of the gravitational potential increases the prevalence of chaotic orbits significantly. They also bear fewer similarities to their associated fractal basin plots. In Figure 4.15, for example, the top left case shows that there are very few regions on the spheroid where local trajectories do *not* diverge. And when these regions were manually inspected, many of them did eventually diverge after very long times ($t \approx 5000$), which confounded attempts to get useful Poincaré sections. This does not necessarily mean those orbits ought to be considered chaotic (λ_{max} would certainly be miniscule), but they do seem to be more divergent than periodic orbits on a comparable massless spheroid.

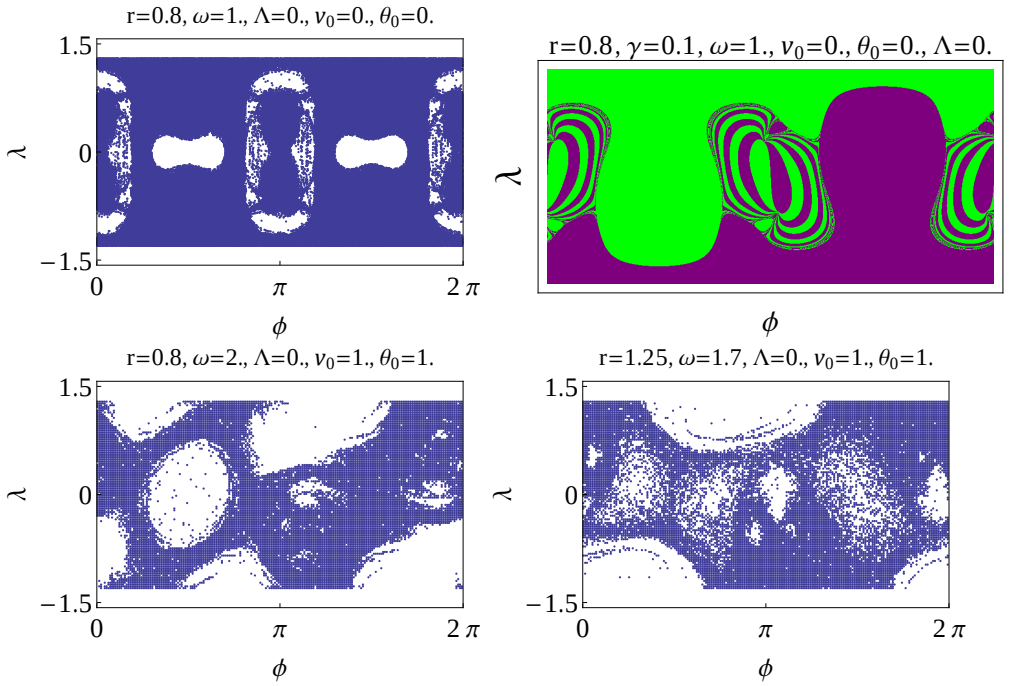


Figure 4.15: Top left: divergence plot of a massive prolate spheroid of aspect ratio $r = 4/5$ and $\Omega = 1$. Dots represent initial conditions for which, after an initial separation of 10^{-12} radians, the local trajectories were at least $10^{-1/2}$ radians distant from one another by $t = 600$. The resolution is $\delta = \pi/500$. On the right is the associated fractal basin plot, with resolution $\delta = \pi/400$ and $\gamma = 0.1$. It bears little resistance to the divergence plot. Bottom: divergence plots for two other massive spheroids, with $\{r, \Omega, v_0, \theta_0\} = \{4/5, 2, 1, 1\}$ on the left and $\{r, \Omega, v_0, \theta_0\} = \{5/4, 1.7, 0, 0\}$ on the right. Both have resolution $\delta = \pi/100$ and were calculated with the same conditions for divergence as above. The corresponding plots for the massless spheroid appear in Figure **??**.

Chapter 5

Conclusions

5.1 Discussion

The system of ellipsoid and slider is uniquely interesting. It is a two-body system in which one of the bodies is a three-dimensional shape and the other is constrained to the surface of that shape. This configuration results in much more complex dynamics than a two-body system in which both bodies can be modeled as points.

The purpose of this project was to simulate the motion of point constrained to slide on the surface of a rotating massive spheroid, with a focus on creating insightful visualizations of that motion. I explored the behavior of the slider on multiple levels:

1. The initial value problem. We showed in chapter 2 that even when gravity is introduced, the path of the slider can be calculated from functions of the variables λ, ϕ, r, Λ, v_0, θ_0, and Ω, where Ω is a ratio of the angular velocity of the ellipsoid ω to its mass density ρ. This is the same number of variables as the zero gravity case, but they reflect a different potential surface.

2. The individual trajectory – or pair of fiducial and adjacent trajectories. Due to the geometry of the ellipsoid, the trace of the Jacobian oscillates as the slider travels over the surface, leading to variable rates of divergence between local trajectories. The trace can also be strobed to generate unique Poincaré sections.

3. The spheroid surface. Patterns emerge when the outcomes of orbits are compared based on their initial coordinates. I showed in chapter 4 that there are regions of chaos and order on the surface, and that these regions reflect fractal basin plots associated with viscosity.

These explorations have led me to conclude the following:

1. The spheroid-slider system is chaotic in both the massless and massive cases. There are more chaotic orbits on massive ellipsoids than massless ones. I have shown that the system is sensitive to initial conditions by measuring the rate at which local trajectories diverge, and I generated Poincaré sections which

show the characteristic features associated with chaos.

2. Chaos is the result of the combination of geometric eccentricity and rotation of the spheroid; these are necessary conditions for chaotic orbits to occur. When the gravitational potential is added to the system, it combines with the centrifugal potential to effectively create hills and valleys on the surface. This potential surface can be described in terms of Ω, a ratio between angular velocity ω and mass density ρ.

3. Fractal geometry can be found in the system when a damping term is added to the initial value problem. In the massless case, viscosity causes the slider to spiral into point attractors in potential basins. Plotting the binary outcomes of the orbits yields fractal basins. In the massive case, the slider can either spiral to a point attractor, or travel in a chaotic or periodic orbit.

5.2 Challenges and Limitations

Drawing generalizable conclusions about the spheroid-slider system is challenging. Each point on the spheroid represents a completely different trajectory, and the spheroid can be stretched and rotated in infinite combinations. And because the equations of motion are so complicated, it is difficult to look to the mathematics of the system for guidance. As a result, the project relied heavily on using *Mathematica's* computing power to simulate a wide variety of initial conditions and look for commonalities.

As a computational project, *Mathematica's* ability to execute the required calculation often presented challenges. Accumulation of numerical error makes chaos difficult to simulate. Ideally, experimental results can bolster theoretical and computational results. However, it is not practical to implement experimental tests of our results at this time; among other difficulties, Earth's gravity would make it impossible. While I am confident that numerical accuracy is high for short time scales, longer runs required several times the machine precision, which increases evaluation time significantly. This can pose a problem when computing Poincaré sections, because in order for a pattern to emerge the orbit may need to run for up to $t = 10^5$. While the most stable periodic orbits are unaffected by small errors, periodic orbits that are local to chaotic orbits may end up getting nudged into the chaotic region as a result of numerical error. This is more likely to occur in the massive case, because the prevalence of chaotic orbits is overall greater, and the equations themselves are more complicated and therefore more prone to error.

Furthermore, while it is tempting to treat chaos as a binary phenomenon–an orbit is either chaotic or it isn't–the spheroid-slider system presents a plethora of edge cases. I encountered initial conditions for which local trajectories diverged, but only very slowly, over thousands of t. It is difficult to say if this is a natural feature of the system or if it is a consequence of numerical error accumulating over long run times. I would suggest that this is a system in which the transition from order to chaos is more akin to a gradient than a phase change, and any line drawn between order and chaos will ultimately be arbitrary.

A significant source of frustration was the coordinate system itself. Latitude and longitude are the most natural parameters for defining the position of the slider, but is pathological at the poles, where the equations are rendered undefined. The coordinate system also obscures the symmetry of the problem in that on a sphere or spheroid, the north and south poles are exactly the same geometrically. This was found to be problematic when calculating the trace of a slider on a static spheroid, because great circles at any angle from the equator should be identical dynamically, yet the changing value of λ causes the trace to oscillate. It still is unclear exactly what, if anything, this mathematical oscillation represents physically, and being able to explain it would go a long way towards proving the Poincaré sections presented in this thesis are valid.

5.3 Future Work

Currently, the spheroid is modeled as rotating rigidly at a constant rate. Future projects could model a massive slider, the motion of which affects the rate of rotation by exchanging its energy. Similarly, the projectile is confined to the surface in this project, but future models could allow the slider to leave the surface.

This project covers the spheroid case only, where two of the three axes of the ellipsoid are equal. Extending the project to include triaxial ellipsoids is possible, but would require using elliptical functions.

The effect of viscosity on the motion of the slider is unclear. In the massless cases, it appears to damp out entirely after a long enough time. However, when gravity is added, it can carry on indefinitely. Future projects could examine how and why this happens.

Control of chaos refers to techniques that seek to either take advantage of chaotic motion to steer the system towards a desired outcome, or to eliminate chaos altogether and encourage periodic behavior. Practical applications include controlling chaos in heart rhythms to prevent cardiac arrest [9] and controlling the motion of satellites [10]. Control of chaos could be applied to the spheroid-slider system by introducing a factor that imparts the minimum amount of force necessary to guide the slider from one point on the spheroid to another.

Appendix A

Dictionary of Symbols

Symbols & Meanings	
Symbol	Meaning
γ	Viscosity
G	Gravitational constant
J	Jacobian matrix of partial derivatives
λ	Latitude
Λ	Angle between x-axis and axis of rotation
ω	Angular velocity
Ω	Nondimensionalized ratio $\frac{\omega}{\sqrt{G\pi\rho}}$
ϕ	Longitude
r	Ratio between the two distinct axes of the spheroid $\left(\frac{a}{c}\right)$
ρ	Mass density $\frac{M}{V}$
σ	Trace of the Jacobian matrix
θ_0	Initial angle of a slider launched with initial velocity v_0
$\mathcal{T}$	Characteristic time $\frac{1}{\sqrt{G\pi\rho}}$
U	Gravitational potential energy
V	Gravitational potential
v_0	Initial velocity in the frame of the spheroid

Appendix B

Mathematica Notebooks

Initial Value Problem

Initialize

In[500]:=
```
$Version
```
Out[500]=
```
10.1.0  for Microsoft Windows (64-bit) (March 24, 2015)
```

In[501]:=
```
Clear["Global`*"]
```

In[502]:=
```
Needs["VariationalMethods`"]
```

Location on Ellipsoid

Define the coordinate system

In[503]:=
```
rTVec[λ_, ϕ_][a_, b_, c_] := {a Cos[λ] Cos[ϕ], b Cos[λ] Sin[ϕ], c Sin[λ]}
```

In[504]:=
```
rTVec[λ_, ϕ_][a_, c_] := rTVec[λ, ϕ][a, a, c] (* spheroid *)
```

Define assumptions to be used later

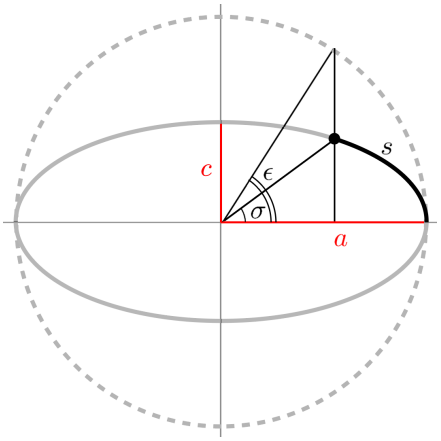
In[505]:=
```
tAssume = {G > 0, m > 0, ρ > 0, -9 < Ω < 9, 0 < a < c, 0 < ϕ[t] < 2 π, -π/2 < λ[t] < π/2};
```

In[506]:=
```
assumeAC = {G > 0, m > 0, ρ > 0, -9 < Ω < 9, 0 < a, 0 < c, 0 < ϕ[t] < 2 π, -π/2 < λ[t] < π/2};
```

In[507]:=
```
assume = tAssume /. x_[t] → x
```
Out[507]=
$$\left\{G > 0,\ m > 0,\ \rho > 0,\ -9 < \Omega < 9,\ 0 < a < c,\ 0 < \phi < 2\,\pi,\ -\frac{\pi}{2} < \lambda < \frac{\pi}{2}\right\}$$

Angle & Angular Velocity Along Ellipse

Put the initial velocities in terms of a magnitude v_0 and heading θ_0



In[508]:=
```
rEllipseVec[ϵ_] := {a Cos[ϵ], c Sin[ϵ]}
```

In[509]:=
```
vEllipseVec = ∂t rEllipseVec[ϵ[t]];
```

In[510]:=
```
dσdt = Solve[Eliminate[{ (* generic ellipse *)
        Tan[σ[t]] == c/a Tan[ϵ[t]], (* central angle σ & eccentric anomaly ϵ *)
        ∂t (Tan[σ[t]] == c/a Tan[ϵ[t]]),
        vs == √(vEllipseVec.vEllipseVec)
      }, {ϵ[t], ϵ'[t]}], σ'[t], Reals] //
    Simplify[#, {c > a > 0, vs > 0, -π/2 < σ[t] < π/2}] &;
```

Eliminate::ifun : Inverse functions are being used by Eliminate,
so some solutions may not be found; use Reduce for complete solution information. >>

In[511]:=
```
dσdt
```

Out[511]=
$$\left\{\left\{\sigma'[t] \to -\frac{\text{vs}\,\text{Cos}[\sigma[t]]^2\left(c^2+a^2\,\text{Tan}[\sigma[t]]^2\right)^{3/2}}{a\,c\,\sqrt{c^4+a^4\,\text{Tan}[\sigma[t]]^2}}\right\},\right.$$
$$\left.\left\{\sigma'[t] \to \frac{\text{vs}\,\text{Cos}[\sigma[t]]^2\left(c^2+a^2\,\text{Tan}[\sigma[t]]^2\right)^{3/2}}{a\,c\,\sqrt{c^4+a^4\,\text{Tan}[\sigma[t]]^2}}\right\}\right\}$$

In[512]:=
```
ωλ[λ_][vλ_][a_, b_, c_] = (σ'[t] /. dσdt[[2]] /. {σ[t] → λ, vs → vλ});
```

In[513]:=
```
vλ[λ_][λDot_][a_, b_, c_] = vλ1 /. First@Solve[ωλ[λ][vλ1][a, b, c] == λDot, vλ1];
```

In[514]:=
```
vλ[λ][λDot][R, R, R] // Simplify[#, assume⋃{R > 0}] & (* spherical special case *)
```

Out[514]=
```
R λDot
```

Angles & Angular Velocities Along Ellipsoid

Latitude & longitude $\{\lambda, \phi\}$ sliding with angular velocities $\{\dot{\lambda}, \dot{\phi}\}$ or with speed & direction $\{v,\theta\}$ with respect to the ellipsoid.

In[515]:=
```
λϕDot[λ_, ϕ_][v_, θ_][a_, b_, c_] :=
 Module[{r, vλ, vϕ, λDot, ϕDot},
  r = √(rTVec[λ, ϕ][a, b, c].rTVec[λ, ϕ][a, b, c]) ;
  vλ = v Cos[θ] ;
  vϕ = v Sin[θ] ;

  λDot = ωλ[λ][vλ][a, b, c] ;
  ϕDot = vϕ / (r Cos[λ]) ;

  {λDot, ϕDot}
 ]
```

In[516]:=
```
vθ[λ_, ϕ_][λDot_, ϕDot_][a_, b_, c_] :=
 Module[{r, vλs, vϕs, v, θ},
  r = √(rTVec[λ, ϕ][a, b, c].rTVec[λ, ϕ][a, b, c]) ;
  vλs = vλ[λ][λDot][a, b, c] ;
  vϕs = ϕDot r Cos[λ] ;
  v = √(vλs^2 + vϕs^2) ;
  θ = ArcTan[vλs, vϕs] ;
  {v, θ}
 ]
```

Viscosity

Define a viscous damping term

In[517]:=
```
αViscosity[λ_, ϕ_][λDot_, ϕDot_][a_, b_, c_] :=
 Module[{v1, θ1, v2, θ2},
  {v1, θ1} = vθ[λ, ϕ][λDot, ϕDot][a, b, c] ;
  v2 = γ v1 ; (* viscous acceleration magnitude *)
  θ2 = θ1 + π ; (* opposite direction *)
  λϕDot[λ, ϕ][v2, θ2][a, b, c]
 ]
```

In[518]:=
```
αViscosity[λ, φ][λDot, ϕDot][a, a, c]
```

Out[518]=
```
{-γ λDot, -γ ϕDot}
```

Apparently, geometrical factors cancel both latitudinally & longitudinally.

Spinning

Define the position vector initially in the non-inertial frame of the spheroid

In[519]:=
```
r1Vec[t_] := {a Cos[λ[t]] Cos[ϕ[t]], a Cos[λ[t]] Sin[ϕ[t]], c Sin[λ[t]]}
```

In[520]:=
```
Rx[θ_] := {{1, 0, 0}, {0, Cos[θ], -Sin[θ]}, {0, Sin[θ], Cos[θ]}}
```

Check that Rx and RotationMatrix are consistent

In[521]:=
```
Rx[θ] == RotationMatrix[θ, {1, 0, 0}]
```

Out[521]=
```
True
```

Define a function that gives the rotation matrix around an angle λ at angular velocity θ

In[522]:=
```
rot[θ_, λ_] := RotationMatrix[θ, {Cos[λ], 0, Sin[λ]}]
```

Get the position vector in the inertial frame

In[523]:=
```
r2Vec[t_] := rot[ωSpin t, λSpin].r1Vec[t] // FullSimplify[#, {λSpin > 0}] &
r2Vec[t] // MatrixForm
```

Out[524]//MatrixForm=

$$\begin{pmatrix} c \sin[2\,\lambda\text{Spin}]\,\sin\left[\frac{t\,\omega\text{Spin}}{2}\right]^2 \sin[\lambda[t]] + a\cos[\lambda[t]]\left(\cos[\phi[t]]\left(\cos[\lambda\text{Spin}]^2 + \cos[t\,\omega\text{Spin}]\right.\right. \\ -c\cos[\lambda\text{Spin}]\sin[t\,\omega\text{Spin}]\sin[\lambda[t]] + a\cos[\lambda[t]]\,(\cos[\phi[t]]\sin[\lambda\text{Spin}]\,\text{Si} \\ c\left(\cos[\lambda\text{Spin}]^2\cos[t\,\omega\text{Spin}] + \sin[\lambda\text{Spin}]^2\right)\sin[\lambda[t]] + a\cos[\lambda[t]]\left(\cos[\phi[t]]\sin[2\,\lambda\text{Spin}\right. \end{pmatrix}$$

Gravity

In[525]:=
```
φOblateInterior =
-G π ρ a^2 c ((1 - (x^2 + y^2 - 2 z^2)/(2 (a^2 - c^2))) 2/Sqrt[a^2 - c^2] ArcSin[Sqrt[(a^2 - c^2)/(a^2 + κ)]] + Sqrt[c^2 + κ]/(a^2 - c^2) (x^2 + y^2)/(a^2 + κ) - 1/(a^2 - c^2) 2 z^2/Sqrt[c^2 + κ]) /. κ → 0;
```

In[526]:=
```
φProlateInterior =
-G π ρ a^2 c ((1 + (x^2 + y^2 - 2 z^2)/(2 (c^2 - a^2))) 2/Sqrt[c^2 - a^2] ArcSinh[Sqrt[(c^2 - a^2)/(κ + a^2)]] - Sqrt[κ + c^2]/(κ + a^2) (x^2 + y^2)/(c^2 - a^2) + 1/(c^2 - a^2) 2 z^2/Sqrt[κ + c^2]) /. κ → 0;
```

Check that these are equivalent expressions

In[527]:=
```
φOblateInterior == φProlateInterior // FullSimplify[#, {a > 0, c > 0}] &
```
Out[527]= True

Get the gravitational potential in the inertial frame (it will be a function of time)

In[528]:=
```
φSurfaceInertial = φProlateInterior /. Thread[{x, y, z} → r2Vec[t]] //
    Simplify[#, Assumptions → tAssume] &;
```

Lagrangian

Calculate

Get the potential energy by multiplying the potential energy per unit mass (φSurfaceInertial) by a mass m

In[529]:=
```
VSurfaceSpinning = m φSurfaceInertial;
```

In[530]:=
```
L = 1/2 m r2Vec'[t].r2Vec'[t] - VSurfaceSpinning //
    Simplify[#, Assumptions → assumeAC] &;
```

Get Euler-Lagrange equations

In[531]:=
```
go = EulerEquations[L, {λ[t], ϕ[t]}, t] // Simplify[#, Assumptions → tAssume] &;
```

Simplify

In[532]:=
```
goo = go // Simplify[#, Assumptions → tAssume] &;
```

Manually add viscosity.

In[533]:=
```
goWithViscosity =
  {λ''[t], ϕ''[t]} ==
    (αViscosity[λ[t], ϕ[t]][λ'[t], ϕ'[t]][a, a, c] + {λ''[t], ϕ''[t]} /. First[Solve[
          go, {λ''[t], ϕ''[t]}]] // Simplify[#, Assumptions → tAssume] &) // Thread;
```

In[534]:=
```
{ωλStart, ωϕStart} = λϕDot[λ0, ϕ0][v0, Θ0][a, a, c];
start = {λ[0] == λ0, ϕ[0] == ϕ0, λ'[0] == ωλStart, ϕ'[0] == ωϕStart} //
    Simplify[#, {tAssume, v0 ≥ 0}] &;
```

In[536]:=
```
IVP = start ~ Join ~ goWithViscosity;
```

In[537]:=
```
IVP // TableForm
```

Out[537]//TableForm=

$$\lambda 0 == \lambda[0]$$
$$\phi 0 == \phi[0]$$
$$\lambda'[0] == \frac{v0\,\text{Cos}[\theta 0]\,\text{Cos}[\lambda 0]^2\,\left(c^2+a^2\,\text{Tan}[\lambda 0]^2\right)^{3/2}}{a\,c\,\sqrt{c^4+a^4\,\text{Tan}[\lambda 0]^2}}$$
$$\phi'[0] == \frac{\sqrt{2}\;v0\,\text{Sec}[\lambda 0]\,\text{Sin}[\theta 0]}{\sqrt{a^2+c^2+(a^2-c^2)\,\text{Cos}[2\,\lambda 0]}}$$
$$\lambda''[t] == -\gamma\,\lambda'[t] + \frac{2\left(-a\,\omega\text{Spin}^2\,\text{Cos}[\lambda[t]]\,\text{Cos}[\phi[t]]\,\text{Sin}[\lambda\text{Spin}]\,\text{Sin}[t\,\omega\text{Spin}]\,(c\,\text{Cos}[\lambda\text{Spin}]\,\text{Cos}[\lambda[t]]\,\text{Sin}[t\,\omega\text{Spin}]+a\,\text{Sin}[\lambda\right.}{}$$
$$\phi''[t] == -\gamma\,\phi'[t] + \text{Sec}[\lambda[t]]^2\left(-\frac{a\,\text{Cos}[\lambda[t]]\left(-6\,a^2\,c\,G\,\pi\,\rho\,\text{ArcSinh}\left[\sqrt{-1+\frac{c^2}{a^2}}\,\right]\text{Cos}[\lambda\text{Spin}]\,\text{Sin}\left[\frac{t\,\omega\text{Spin}}{2}\right]\right)\left(-\text{Cos}\left[\frac{t\,\omega\text{Spin}}{2}\right]\text{Cos}[\phi[t]]+\text{Sin}[\lambda\text{Spin}]\,\text{Sin}\left[\frac{t\,\omega\text{Spin}}{2}\right]\text{Sin}[\phi[t]]\right.}{}\right.$$

Nondimensionalize

Set Up

Define substitutions to cancel dimensional quantities out of the equations.

In[538]:=
```
varSub = {
    λ → (λN[# / T] &),
    ϕ → (ϕN[# / T] &),
    t → tN T,
    vC → (vCN[# / T] c^2 / T^2 &) (* potential *),
    vG → (vGN[# / T] c^2 / T^2 &)
   };
paramSub = {
    a → rN c,
    v0 → v0N c / T,
    ωSpin → ωSpinN / T,

    λSpin → λSpinN,
    γ → γN / T(* viscosity *)
   };
choiceSub = {T → 1 / Sqrt[G π ρ]};
```

In[541]:=
```
varsN = {λN[0], ϕN[0], λN'[0], ϕN'[0], λN''[tN], ϕN''[tN]};
```

In[542]:=
```
rVec[t] = r2Vec[t] /. {ωSpin → ωSpinN / T, λSpin → λSpinN, γ → γN / T};
```

Computation

In[543]:=
```
IVPN =
  IVP //. varSub //. paramSub //. choiceSub // Simplify[#, Assumptions → {assumeAC}] &;

varsN = {λN[0], ϕN[0], λN'[0], ϕN'[0], λN''[tN], ϕN''[tN]};

IVPN2 =
  varsN == (varsN /. First@Solve[IVPN, varsN] //
       Simplify[#, Assumptions → {assumeAC}] &) // Thread;

IVPN2 // TableForm
```

Manually Simplified Form (Λ=0)

In[449]:=
$$
\begin{aligned}
&\text{IVPN2} = \{\\
&\quad \lambda N[0] == \lambda 0,\\
&\quad \phi N[0] == \phi 0,\\
&\quad \lambda N'[0] == \frac{v0N \,\text{Cos}[\theta 0]\, \text{Cos}[\lambda 0]^2 \left(1 + rN^2 \,\text{Tan}[\lambda 0]^2\right)^{3/2}}{rN \sqrt{1 + rN^4 \,\text{Tan}[\lambda 0]^2}},\\
&\quad \phi N'[0] == \frac{\sqrt{2}\; v0N \,\text{Sec}[\lambda 0]\, \text{Sin}[\theta 0]}{\sqrt{1 + rN^2 + \left(-1 + rN^2\right) \text{Cos}[2\,\lambda 0]}},\\
&\quad \lambda N''[tN] ==\\
&\qquad -\gamma N\, \lambda N'[tN] + \frac{\text{Sin}[2\,\lambda N[tN]]}{2\left(\text{Cos}[\lambda N[tN]]^2 + rN^2\, \text{Sin}[\lambda N[tN]]^2\right)} \left(-\frac{rN^2\left(2 + rN^2\right)}{\left(1 - rN^2\right)^{3/2}} \text{ArcCsch}\left[\frac{rN}{\sqrt{1 - rN^2}}\right]\right.\\
&\qquad\quad \left(-1 + 3\,\text{Cos}[tN\,\omega SpinN]^2 + \frac{3\,rN}{\left(2 + rN^2\right)} \left(rN \,\text{Cos}[2\,\phi N[tN]]\, \text{Sin}[tN\,\omega SpinN]^2 + \right.\right.\\
&\qquad\qquad \left.\left. 2\,\text{Cot}[2\,\lambda N[tN]]\, \text{Sin}[2\,tN\,\omega SpinN]\, \text{Sin}[\phi N[tN]]\right)\right) + \frac{1}{2\left(-1 + rN^2\right)}\\
&\qquad\quad \left(2 - 7\,rN^2 + 2\,rN^4 - \left(2 - 3\,rN^2 + rN^4\right) \omega SpinN^2 + rN^2 \left(-1 - \omega SpinN^2 + rN^2 \left(-2 + \omega SpinN^2\right)\right)\right.\\
&\qquad\qquad \text{Cos}[2\,\phi N[tN]] - 2\left(1 + 2\,rN^2\right) \text{Cos}[2\,tN\,\omega SpinN] \left(1 + rN^2\, \text{Sin}[\phi N[tN]]^2\right) -\\
&\qquad\qquad 4\left(rN + 2\,rN^3\right) \text{Cot}[2\,\lambda N[tN]]\, \text{Sin}[2\,tN\,\omega SpinN]\, \text{Sin}[\phi N[tN]]) - \left(-1 + rN^2\right)\\
&\qquad\quad \left.\left.\lambda N'[tN]^2 - rN\, \phi N'[tN] \left(2\,\omega SpinN \,\text{Cos}[\phi N[tN]]\, \text{Cot}[\lambda N[tN]] + rN\, \phi N'[tN]\right)\right),\right.\\
&\quad \phi N''[tN] == -\gamma N\, \phi N'[tN] - \frac{1}{rN\left(1 - rN^2\right)^{3/2}} \left(3\,rN^2 \,\text{ArcSinh}\left[\sqrt{\frac{1}{rN^2} - 1}\right]\right.\\
&\qquad \left(rN \,\text{Sin}[tN\,\omega SpinN]^2\, \text{Sin}[2\,\phi N[tN]] + \text{Cos}[\phi N[tN]]\, \text{Sin}[2\,tN\,\omega SpinN]\, \text{Tan}[\lambda N[tN]]\right) +\\
&\qquad \sqrt{1 - rN^2} \left(-2\,rN^3\, \text{Sin}[tN\,\omega SpinN]^2\, \text{Sin}[2\,\phi N[tN]] +\right.\\
&\qquad\quad \text{Cos}[\phi N[tN]] \left(rN\left(-1 + \left(-1 + rN^2\right) \omega SpinN^2 + \text{Cos}[2\,tN\,\omega SpinN]\right) \text{Sin}[\phi N[tN]] -\right.\\
&\qquad\qquad \left.\left.\left(1 + 2\,rN^2\right) \text{Sin}[2\,tN\,\omega SpinN]\, \text{Tan}[\lambda N[tN]]\right)\right) -\\
&\qquad \left.\left.2\left(1 - rN^2\right)^{3/2} \lambda N'[tN] \left(\omega SpinN \,\text{Cos}[\phi N[tN]] + rN\, \text{Tan}[\lambda N[tN]]\, \phi N'[tN]\right)\right)\right.\\
&\};
\end{aligned}
$$

Sphere

Calculate

```
IVPNsphere = varsN == Limit[Table[IVPN2[[i, 2]], {i, 1, 6}], rN → 1] // Thread //
  Simplify[#, {-π/2 < λ0 < π/2, 0 < ϕ0 < 2 π}] &
```

$$\{\lambda 0 == \lambda N[0],\ \phi 0 == \phi N[0],\ v0\,\text{Cos}[\theta 0] == \lambda N'[0],\ v0\,\text{Sec}[\lambda 0]\,\text{Sin}[\theta 0] == \phi N'[0],$$

$$\lambda N''[tN] == \frac{1}{4}\Big(\omega\text{SpinN}^2\Big(-2\,\text{Cos}[2\,\lambda N[tN]]\,\text{Cos}[\phi N[tN]]\,\text{Sin}[2\,\lambda\text{SpinN}] + \frac{1}{4}\,(-2 + 6\,\text{Cos}[2\,\lambda\text{SpinN}] + \text{Cos}[2\,\lambda\text{SpinN} - 2\,\phi N[tN]] + 2\,\text{Cos}[2\,\phi N[tN]] + \text{Cos}[2\,(\lambda\text{SpinN} + \phi N[tN])])\,\text{Sin}[2\,\lambda N[tN]]\Big) - 4\,\gamma N\,\lambda N'[tN] - 4\,\omega\text{SpinN}\,\Big(2\,\text{Cos}[\lambda\text{SpinN}]\,\text{Cos}[\lambda N[tN]]^2\,\text{Cos}[\phi N[tN]] + \text{Sin}[\lambda\text{SpinN}]\,\text{Sin}[2\,\lambda N[tN]]\Big)\,\phi N'[tN] - 2\,\text{Sin}[2\,\lambda N[tN]]\,\phi N'[tN]^2\Big),$$

$$\omega\text{SpinN}^2\,\Big(\text{Cos}[\lambda\text{SpinN}]^2\,\text{Sin}[2\,\phi N[tN]] + \text{Sin}[2\,\lambda\text{SpinN}]\,\text{Sin}[\phi N[tN]]\,\text{Tan}[\lambda N[tN]]\Big) + 4\,\lambda N'[tN]\,(\omega\text{SpinN}\,\text{Cos}[\lambda\text{SpinN}]\,\text{Cos}[\phi N[tN]] + \omega\text{SpinN}\,\text{Sin}[\lambda\text{SpinN}]\,\text{Tan}[\lambda N[tN]] + \text{Tan}[\lambda N[tN]]\,\phi N'[tN]) == 2\,(\gamma N\,\phi N'[tN] + \phi N''[tN])\}$$

Quick

```
IVPNsphere = {
   λ0 == λN[0],
   ϕ0 == ϕN[0],
   v0N Cos[θ0] == λN'[0],
   v0N Sec[λ0] Sin[θ0] == ϕN'[0],
   ωSpinN^2 Cos[λN[tN]] Cos[ϕN[tN]]^2 Sin[λN[tN]] ==
    γN λN'[tN] + 2 ωSpinN Cos[λN[tN]]^2 Cos[ϕN[tN]] ϕN'[tN] +
     Cos[λN[tN]] Sin[λN[tN]] ϕN'[tN]^2 + λN''[tN], ωSpinN^2 Cos[ϕN[tN]] Sin[ϕN[tN]] +
     2 λN'[tN] (ωSpinN Cos[ϕN[tN]] + Tan[λN[tN]] ϕN'[tN]) == γN ϕN'[tN] + ϕN''[tN]};
```

Visualizations

Initialize

```
$Version
"10.1.0  for Microsoft Windows (64-bit) (March 24, 2015)"
Clear["Global`*"]
Needs["VariationalMethods`"]

meshSpec[ϵ_] := {9, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 - ϵ} π/6}

(* corrects Mathematica drawing bug *)
label[r_, Ω_, Λ_, γ_] :=
  "a/c = " <> ToString[r // N // NumberForm[#, 4] &] <>
   ", ω/√(G π ρ) = " <> ToString[Ω // N // NumberForm[#, 4] &] <>
   ", Λ = " <> ToString[Λ // N // NumberForm[#, 4] &];
```

IVP

BE SURE TO RUN THE IVP NOTEBOOK FIRST AND OBTAIN IVPN

Poincare Sections

Jacobian

In[186]:=
```
yVec = {λN[tN], ϕN[tN], λN'[tN], ϕN'[tN]};
```

Get a vector fVec that consists of λ', ϕ', λ'', and ϕ''

In[188]:=
```
fVec = {λN'[tN], ϕN'[tN]} ⋃ {λN''[tN], ϕN''[tN]} /.
   First@Solve[IVPN2, {λN''[tN], ϕN''[tN]}];
```

Get the Jacobian by taking the partial derivative of every element in fVec with respect to every element in yVec

In[189]:=
```
jacobian =
  D[fVec, {yVec}] // Simplify[#, {assumeAC, -π/2 < λN[tN] < π/2, 0 < ϕN[tN] < 2 π}] &;
```

Get the trace

In[191]:=
```
trJ = Tr[jacobian] // FullSimplify
```

Out[191]=
$$-2\,\gamma N + \frac{2\left(1-\left(-1+rN^2\right)\text{Cos}[2\,\lambda N[tN]]\right)\text{Tan}[\lambda N[tN]]\,\lambda N'[tN]}{\text{Cos}[\lambda N[tN]]^2 + rN^2\,\text{Sin}[\lambda N[tN]]^2}$$

Check that the matrix has the expected dimensions (4x4)

In[192]:=
```
Dimensions[jacobian]
```

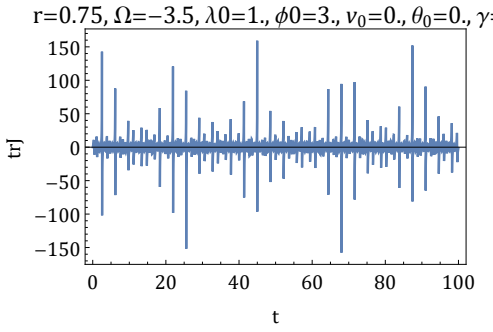
Out[192]=
```
{4, 4}
```

Plots of the Trace

"sol" refers to numerical solutions to the IVP

In[432]:=
```
Plot[trJ /. sol /. parameters /. initials, {tN, 0, 100
 }, Frame → True, FrameLabel → {"t", "trJ"},
 FrameTicks → {{Automatic, None}, {Automatic, None}},
 PlotLabel → "r=" <> ToString[rN /. parameters // N] <>
   ", Ω=" <> ToString[ωSpinN /. parameters // N] <> ", λ0=" <>
   ToString[λ0 /. initials // N] <> ", ϕ0=" <> ToString[ϕ0 /. initials // N] <>
   ", v0=" <> ToString[v0N /. initials // N] <> ", θ0=" <>
   ToString[θ0 /. initials // N] <> ", γ=" <> ToString[γN /. parameters],
 LabelStyle → Directive[FontSize → 14, FontFamily -> "Cambria"], PlotRange → All]
```

Out[432]=



Function

2D sections strobed on trace

```
sectiontrace[acRatio_, λStart_, ϕStart_, vStart_, θStart_][
   spinRate_, spinAngle_, γ_][num_, color_, wp_] :=
 Module[{tsection, initial, checkZero, greaterthancheck},
  (*set of initial conditions we want to solve for*)

  parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → γ};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
  tsection = num;

  {sol, {points}} = Reap@NDSolve[
     {IVPN2 /. parameters /. initials, WhenEvent[Evaluate[(trJ) == 0 ⋀ D[trJ, tN] > 0],
        Sow[{Mod[ϕN[tN], 2 π], λN[tN]}]]} /. parameters,
     {λN[tN], ϕN[tN], λN'[tN], ϕN'[tN]}, {tN, tsection},
     WorkingPrecision → wp, MaxSteps → Infinity];

  ListPlot[points, PlotRange → {{0, 2 π}, {-π/2, π/2}}, PlotStyle → color,
   Frame -> True,
   FrameTicks → {{{-π/2, 0, π/2}, None}, {{0, π, 2 π}, None}}, FrameLabel → {ϕ, λ},
   PlotLabel → "r=" <> ToString[acRatio /. parameters // N] <> ", Ω=" <> ToString[
      spinRate /. parameters // N] <> ", λ0=" <> ToString[λStart /. initials // N] <>
     ", ϕ0=" <> ToString[ϕStart /. initials // N] <> ", v₀=" <> ToString[vStart // N] <>
     ", θ₀=" <> ToString[θStart // N] <> ", γ=" <> ToString[γ // N],
   LabelStyle → Directive[FontSize → 14, FontFamily -> "Cambria"]]
 ]
```

In[250]:=
```
vVec[tN_] =
  r2Vec'[t] /. λ[t] → λN[tN] /. ϕ[t] → ϕN[tN] /. λSpin → λSpinN /. ωSpin → ωSpinN /.
        ϕ'[t] -> ϕN[tN] /. λ'[t] → λN[tN] /. a → rN c /. c → 1 /. t → tN // FullSimplify;
```

3D sections strobed on trace

```
In[335]:= sectiontrace3D[acRatio_, λStart_, ϕStart_, vStart_, θStart_][
            spinRate_, spinAngle_, γ_][num_, color_, wp_] :=
          Module[{tsection, initial, checkZero, greaterthancheck},
           (*set of initial conditions we want to solve for*)

           parameters =
            {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → γ, λSpinN → 0};
           initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
           tsection = num;

           {sol, {points}} = Reap@NDSolve[
              {
                IVPN2 /. parameters /. initials,
                WhenEvent[Evaluate[(trJ) == 0 ⋀ D[trJ, tN] > 0], Sow[
                  {
                    Mod[ϕN[tN], 2 π], λN[tN], Log[√vVec[tN].vVec[tN] ]
                   } /. parameters]]
               } /. parameters,
              {λN[tN], ϕN[tN], λN'[tN], ϕN'[tN]},
              {tN, tsection}, WorkingPrecision → wp, MaxSteps → Infinity];

           ListPointPlot3D[
            points
            , PlotRange → {{0, 2 π}, {-π/2, π/2}, {Min@points[[All, 3]], Max@points[[All, 3]]}}
            , PlotStyle → color
            , Boxed → True
            , Ticks → {{0, π, 2 π}, {-π/2, 0, π/2}, Automatic}
            , AxesLabel → {ϕ, λ, "Log[velocity]"}
            , PlotLabel → "r=" <> ToString[acRatio /. parameters // N] <>
              ", Ω=" <> ToString[spinRate /. parameters // N] <>
              ", λ0=" <> ToString[λStart /. initials // N] <>
              ", ϕ0=" <> ToString[ϕStart /. initials // N] <>
              ", v0=" <> ToString[vStart // N] <>
              ", θ0=" <> ToString[θStart // N] <>
              ", γ=" <> ToString[γ // N]
            , LabelStyle → Directive[FontSize → 18, FontFamily → "Cambria"]
           ]
          ]
```

2D sections strobed on revolution

```
section[acRatio_, λStart_, ϕStart_, vStart_, θStart_][spinRate_, spinAngle_, γ_][
  num_, color_, wp_] := Module[{tsection, initial, checkZero, greaterthancheck},
  (*set of initial conditions we want to solve for*)


  parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → γ};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
  tsection = num;

  {sol, {points}} = Reap@NDSolve[
     {IVPN2 /. parameters /. initials, WhenEvent[Evaluate[Mod[tN, 2 π / spinRate] == 0],
        Sow[{Mod[ϕN[tN], 2 π], λN[tN]}]]} /. parameters,
     {λN[tN], ϕN[tN], λN'[tN], ϕN'[tN]}, {tN, tsection},
     WorkingPrecision → wp, MaxSteps → Infinity];

  ListPlot[points, PlotRange → {{0, 2 π}, {-π / 2, π / 2}}, PlotStyle → color,
   Frame -> True,
   FrameTicks → {{{-π / 2, 0, π / 2}, None}, {{0, π, 2 π}, None}}, FrameLabel → {ϕ, λ},
   PlotLabel → "r=" <> ToString[acRatio /. parameters // N] <> ", Ω=" <> ToString[
      spinRate /. parameters // N] <> ", λ0=" <> ToString[λStart /. initials // N] <>
     ", ϕ0=" <> ToString[ϕStart /. initials // N] <> ", v₀=" <> ToString[vStart // N] <>
     ", θ₀=" <> ToString[θStart // N] <> ", γ=" <> ToString[γ // N],
   LabelStyle → Directive[FontSize → 14, FontFamily -> "Cambria"]]
 ]
```

3D sections strobed on revolution

In[287]:=
```
section3D[acRatio_, λStart_, ϕStart_, vStart_, θStart_][spinRate_, spinAngle_, γ_][
  num_, color_, wp_] := Module[{tsection, initial, checkZero, greaterthancheck},
  (*set of initial conditions we want to solve for*)

  parameters =
   {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → γ, λSpinN → 0};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
  tsection = num;

  {sol, {points}} = Reap@NDSolve[
     {
       IVPN2 /. parameters /. initials,
       WhenEvent[Evaluate[Mod[tN, 2 π / spinRate] == 0], Sow[
          {
            Mod[ϕN[tN], 2 π], λN[tN], Log[√(vVec[tN].vVec[tN])]
           } /. parameters]]
      } /. parameters,
     {λN[tN], ϕN[tN], λN'[tN], ϕN'[tN]},
     {tN, tsection}, WorkingPrecision → wp, MaxSteps → Infinity];

  ListPointPlot3D[
   points
   , PlotRange → {{0, 2 π}, {-π/2, π/2}, {Min@points[[All, 3]], Max@points[[All, 3]]}}
   , PlotStyle → color
   , Boxed → True
   , Ticks → {{0, π, 2 π}, {-π/2, 0, π/2}, Automatic}
   , AxesLabel → {ϕ, λ, "Log[velocity]"}
   , PlotLabel → "r=" <> ToString[acRatio /. parameters // N] <>
     ", Ω=" <> ToString[spinRate /. parameters // N] <>
     ", λ0=" <> ToString[λStart /. initials // N] <>
     ", ϕ0=" <> ToString[ϕStart /. initials // N] <>
     ", v₀=" <> ToString[vStart // N] <>
     ", θ₀=" <> ToString[θStart // N]
   , LabelStyle → Directive[FontSize → 18, FontFamily → "Cambria"]
  ]
 ]
```

Manipulator (Relative Palette)

Set Up

Format the labels as desired.

```
Solve[ωSpin == ωSpinN / T /. choiceSub, ωSpinN]
```

$$\left\{\left\{\omega\text{SpinN} \to \frac{\omega\text{Spin}}{\sqrt{\pi}\ \sqrt{\text{G}\,\rho}}\right\}\right\}$$

```
label2[r_, Ω_, Λ_, γ_, v0_, θ0_] :=
  "rn = " <> ToString[r // N // NumberForm[#, 4] &] <>
   ", ω/√(G π ρ) = " <> ToString[Ω // N // NumberForm[#, 4] &] <>
   ", v0 = " <> ToString[v0 // N // NumberForm[#, 4] &] <>
   ", θ0 = " <> ToString[θ0 // N // NumberForm[#, 4] &] <>
   ", Λn = " <> ToString[Λ // N // NumberForm[#, 4] &] <>
   ", γn = " <> ToString[γ // N // NumberForm[#, 4] &];
```

Define a function that will generate images of a spheroid with color-coded potential surface

```
framecolor[acRatio_, spinRatio_, spinAngle_, viscosity_][
  λStart_, ϕStart_, vStart_, θStart_] :=
 Module[{IVPN2, φV, params, Vs, vG, aM = 1, ϵ = 10^-2, ϵ0 = 10^-3, VMin,
    VMax, sliderPlotPoints = 20, staticPlotPoints = 20, showAll,
    viewPoint = acRatio RotationMatrix[90 °, {1, 0, 0}].{1.3, 2.4, 2.}
   },

   parameters =
    {rN → acRatio, ωSpinN → spinRatio, λSpinN → spinAngle, γN → viscosity};
   initials = {λ0 → λStart, ϕ0 → ϕStart, v0 → vStart, θ0 → θStart};

   If[acRatio > 1, V[λN_, ϕN_] = totalVON /. parameters,
    V[λN_, ϕN_] = totalVPN /. parameters];

   VSample = {(* find local minima & maxima *)
       FindMinimum[{V[λN, ϕN], -π/2 < λN < π/2, 0 < ϕN < 2 π}, {{λN, #[[1]]}, {ϕN, #[[2]]}}][[
        1]], FindMaximum[{V[λN, ϕN], -π/2 < λN < π/2, 0 < ϕN < 2 π},
         {{λN, #[[1]]}, {ϕN, #[[2]]}}][[1]]
      } & /@ {
      {ϵ0, ϵ0}, {ϵ0, ϵ0 + π/2}, {ϵ0, ϵ0 + π}, {ϵ0, ϵ0 + 3 π/2},
      {-π/2 + ϵ0, ϵ0}, {+π/2 + ϵ0, ϵ0}
     };
   {VMin, VMax} = {Min@VSample, Max@VSample};
   (* find global min & max? *)

   surfacePlot = ParametricPlot3D[rTVec[λ, ϕ][acRatio, acRatio, 1]
     , {λ, -π/2, π/2}, {ϕ, 0, 2 π}
     , ColorFunction → Function[{x, y, z, λ, ϕ}, Hue[0.85 (V[λ, ϕ] - VMin)/(VMax - VMin)]]]
```

```
    , ColorFunctionScaling → False
    , PlotRange → All
    , Lighting → "Neutral"
    , Boxed → False
    , Axes → False
    , Mesh → False
    , SphericalRegion → True
    , ImageSize → Large
    , PlotLabel → Style[label[acRatio, spinRatio, spinAngle, viscosity],
      21, Black, FontFamily → "Cambria"]
    , Background → White
    , PlotPoints → 100
   ];
  axisPlot = Graphics3D[{
     Cyan, Arrowheads[0.07], Arrow@
      Tube[{{0, 0, 0}, 1.2 {Cos[spinAngle], 0, Sin[spinAngle]}}, Scaled[0.006]]
    }];

  showAll = Show[surfacePlot, axisPlot]
 ] // Quiet
```

Define a function that makes a contour plot

```
framecolor2D[acRatio_, spinRatio_, spinAngle_, viscosity_][contours_, lines_] :=
 Module[{IVPN2, φV, params, vG, vVec, g0, g1, g2, g3, g4, aM = 1, ϵ = 10^-2, ϵ0 = 10^-3,
   VMin, VMax, sliderPlotPoints = 20, staticPlotPoints = 20, showAll,
   viewPoint = acRatio RotationMatrix[90 °, {1, 0, 0}].{1.3, 2.4, 2.}
  },

  parameters =
   {rN → acRatio, ωSpinN → spinRatio, λSpinN → spinAngle, γN → viscosity};
  V[λN_, ϕN_] = totalV /. parameters;

  VSample = {(* find local minima & maxima *)
      FindMinimum[{V[λN, ϕN], -π/2 < λN < π/2, 0 < ϕN < 2 π}, {{λN, #[[1]]}, {ϕN, #[[2]]}}][[
       1]], FindMaximum[{V[λN, ϕN], -π/2 < λN < π/2, 0 < ϕN < 2 π},
        {{λN, #[[1]]}, {ϕN, #[[2]]}}][[1]]
     } & /@ {
     {ϵ0, ϵ0}, {ϵ0, ϵ0 + π/2}, {ϵ0, ϵ0 + π}, {ϵ0, ϵ0 + 3 π/2},
     {-π/2 + ϵ0, ϵ0}, {+π/2 + ϵ0, ϵ0}
    };
  {VMin, VMax} = {Min@VSample, Max@VSample};
  (* find global min & max? *)

  surfacePlot = ContourPlot[V[λ, ϕ], {ϕ, 0, 2 π}, {λ, -π/2, π/2}
    , ContourLines → lines
    , Contours → contours
    , ColorFunction → Function[{λ, ϕ}, Hue[0.85 * λ]]
    , FrameTicks → {{{-π/2, 0, π/2}, None}, {{0, Pi, 2 Pi}, None}}
    , FrameTicksStyle → Directive[Black, 15, FontFamily → "Cambria"]
    , FrameLabel → {ϕ, λ}
    , LabelStyle → Directive[19, Black, FontFamily → "Cambria"]
    , PlotLabel → Style[label[acRatio, spinRatio, spinAngle, viscosity],
      17, Black, FontFamily → "Cambria"]
    , AspectRatio → Automatic];

  showAll = Show[surfacePlot]
 ] // Quiet
```

3D Sliding Paths

Manipulator

Define a function to display orbits on a spheroid of desired proportions

In[434]:=
```
frame[acRatio_, spinRate_, spinAngle_, viscosity_][
   λStart_, ϕStart_, vStart_, θStart_][tMax_, tube_, color_] :=
```

```
Module[{ (*λs,ϕs*), surfacePlot, pathPlot, axisPlot},

 parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → viscosity};

 initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
 {λs, ϕs} = Check[
   NDSolveValue[IVPN2 /. parameters /. initials, {λN, ϕN}, {tN, 0, tMax}], $Failed];

 surfacePlot = ParametricPlot3D[rTVec[λ, ϕ][acRatio, acRatio, 1]
    , {λ, -π/2, π/2}, {ϕ, 0, 2 π}
    , PlotRange → All
    , PlotStyle → White
    , Lighting → "Neutral"
    , Boxed → False
    , Axes → False
    , Mesh → meshSpec[]
    , SphericalRegion → True
    , ImageSize → Large
    , PlotLabel → Style[label[acRatio, spinRate, spinAngle, viscosity], 16, White]
    , Background → White
  ];

 pathPlot = ParametricPlot3D[rTVec[λs[tN], ϕs[tN]][acRatio, acRatio, 1]
    , {tN, 0, tMax}
    , Axes → False
    , PlotStyle → If[tube, Directive[Cyan, Tube[0.01]], Black, Black]
    , ColorFunction → If[color, Function[{x, y, z, t}, Hue[0.85 t]]]
    (*If[color,Function[{x,y,z,t},Hue[0.85*tMax/1000 t]]]*)
    , Boxed → True
    , ViewVertical → {1, 0, 0}
    , SphericalRegion → True
    , ImageSize → Large
  ];

 axisPlot = Graphics3D[{
     Cyan, Arrowheads[0.07], Arrow@
      Tube[{{0, 0, 0}, 1.2 { Cos[spinAngle], 0, Sin[spinAngle]}}, Scaled[0.006]]
   }];

 If[λs == $Failed ⋁ ϕs == $Failed, Show[surfacePlot, axisPlot],
  Show[surfacePlot, pathPlot, axisPlot], Show[surfacePlot, pathPlot, axisPlot]]
]
```

Use the manipulator to adjust the spheroid in real time

```
With[{δ = 10^-9, δ2 = 5 × 10^-2},
 Manipulate[frame[rN, ωSpinN, λSpinN, γN][λ0, ϕ0, v0, θ0][tMax, tube, color]
   , {{rN, 0.8, "rn"}, δ2, 3}
   , {{ωSpinN, 2.25, "Ωn"}, -7, 7}
   , {{λSpinN, 0, "Λn"}, 0, π/2}
   , {{γN, 0.07, "γn"}, 0, 1}
   , Delimiter
   , {{λ0, -0.3, "λn0"}, -π/2 + δ, π/2 - δ}
   , {{ϕ0, 0, "ϕn0"}, 0, 2 π}
   , {{v0, 0, "Vn0"}, 0, 9}
   , {{θ0, 1, "θn0"}, δ, 2 π - δ}
   , Delimiter
   , {{tMax, 10, "max tn"}, 1, 1000}
   , Delimiter
   , {{tube, False, "tube path"}, {True, False}}
   , {{color, False, "color path"}, {True, False}}
 ]
]
```

MaxLyap

Algorithm for the maximum Lyapunov exponent

In[17]:=

```
logSep[t_][logsep0_, λ_] := logsep0 + λ t

maxLyap[acRatio_, spinAngle_, spinRate_, viscosity_][δ_,
       T_, Δt_, λStart_, ϕStart_, vStart_, θStart_][Δtmin_, Δtwidth_][
    plotPoints_, Tstart_, Tend_][showFit_][zoom_] := Module[{},

  (*replacement rules*)
  parameters = {
    rN → acRatio,
    λSpinN → spinAngle,
    ωSpinN → spinRate,
    γN → viscosity
   };

  (*numerically integrate a fiducial and adjacent trajectory*)
  initialsf = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
  fiducialOrbit[tt_] =
   {λf, ϕf} = NDSolveValue[IVPN2 /. parameters /. initialsf, {λN, ϕN}, {tN, 0, T},
      MaxSteps → Infinity, WorkingPrecision → MachinePrecision] /. tN → tt;
```

```
initialsa = {λ0 → λStart + 10^δ, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};
adjacentOrbit[tt_] =
 {λa, ϕa} = NDSolveValue[IVPN2 /. parameters /. initialsa, {λN, ϕN}, {tN, 0, T},
    MaxSteps → Infinity, WorkingPrecision → MachinePrecision] /. tN → tt;

(*display the spheroid*)
surfacePlot = ParametricPlot3D[rTVec[λ, ϕ][acRatio, acRatio, 1]
  , {λ, -π/2, π/2}, {ϕ, 0, 2 π}
  , PlotRange → All
  , PlotStyle → White
  , Lighting → "Neutral"
  , Boxed → False
  , Axes → False
  , Mesh -> None
  , SphericalRegion → True
  , ImageSize -> Medium
  , PlotLabel → Style[label[acRatio, spinRate, spinAngle, viscosity], 16, White]
  , Background → White
 ];

(*display the rotation axis*)
axisPlot = Graphics3D[{
   Cyan, Arrowheads[0.07], Arrow@
    Tube[{{0, 0, 0}, 1.3 { Cos[spinAngle], 0, Sin[spinAngle]}}, Scaled[0.006]]
  }];

(*plot the fiducial orbit*)
pathPlotF = ParametricPlot3D[rTVec[λf[tN], ϕf[tN]][acRatio, acRatio, 1]
  , {tN, 0, Tend}
  , Axes → False
  , PlotStyle → Directive[Cyan, Tube[0.01]]
  , Boxed → True
  , ViewVertical → {1, 0, 0}
  , SphericalRegion → True
  , ImageSize → Medium
  , PlotPoints → plotPoints
 ];
(*plot the adjacent orbit*)
pathPlotA = ParametricPlot3D[rTVec[λa[tN], ϕa[tN]][acRatio, acRatio, 1]
  , {tN, 0, Tend}
  , Axes → False
  , PlotStyle → Directive[Pink, Tube[0.01]]
  , Boxed → True
```

```
  , ViewVertical → {1, 0, 0}
  , SphericalRegion → True
  , ImageSize → Medium
  , PlotPoints → plotPoints
 ];

(*sample δr and take the log*)
δOrbit[t_] = {λf[t], ϕf[t]} - {λa[t], ϕa[t]};
δOrbitSampled = Table[δOrbit[t], {t, 0, T, Δt}];
ΔtSampled = Range[0, T, Δt];
separations = Norm /@ δOrbitSampled;
logSeparations = Log[10, separations];
tlogsep = Transpose[{ΔtSampled, logSeparations}];
tLogSepPlot = ListPlot[tlogsep, Frame → True, FrameLabel → {"t", "log δr"},
  LabelStyle → Directive[FontFamily → "Cambria", FontSize → 20],
  PlotRange → {{0, zoom}, All}];
(*lineStyle={Green,Dashed};*)
line = Line[{{Tend, Min@logSeparations}, {Tend, Max@logSeparations}}];

(*define the area of fit*)
Δtmax = Δtwidth + Δtmin;
selectedData = Select[tlogsep, Δtmin ≤ #[[1]] ≤ Δtmax &];
nlm = NonlinearModelFit[selectedData, logSep[t][logsep0, λ], {logsep0, λ}, t];
λEstimate = nlm["ParameterTableEntries"][[2, 1]];
λStandardError = nlm["ParameterTableEntries"][[2, 2]];
R2 = nlm["AdjustedRSquared"];
fitPlot = Plot[nlm[t], {t, 0, Last@ΔtSampled},
  Frame → True, FrameLabel → {"t", "log δr"}, PlotStyle → Red];

If[tlogsep[[2]] > 0, Print[", δLat = " <> ToString[NumberForm[δ // N, 3]] <>
   ", lat0=" <>
   ToString[NumberForm[λStart // N, 3, NumberFormat → (Row[{#1, "e", #3}] &)]] <>
   ", long0=" <> ToString[NumberForm[ϕStart // N, 3,
     NumberFormat → (Row[{#1, "e", #3}] &)]] <>
   ", {rx,ry,rz}=" <> ToString[NumberForm[{acRatio, acRatio, 1} // N, 3]]]];

resultText = "R̄² = " <> ToString[NumberForm[R2, 3]] <>
  ", λ = " <> ToString@NumberForm[λEstimate, 3] <> " ± " <>
  ToString@NumberForm[λStandardError, 1](*<>result*);
fitPlotEpilog = {Text[Style[resultText, FontFamily → "Arial", FontSize → 16],
   Scaled[{0.7, 0.1}]]};

lyapunovPlot = If[showFit,
```

```
  Show[tLogSepPlot, fitPlot,
   PlotLabel → Style[
     "Δt = " <> ToString[NumberForm[Δt // N, 3]] <>
      ", δ = " <> ToString[NumberForm[δ // N, 3]] <>
      ", r=" <> ToString[NumberForm[acRatio // N, 3]] <>
      ", Ω=" <> ToString[NumberForm[spinRate // N, 3]] <>
      ", λ0=" <> ToString[NumberForm[λStart // N, 4]] <>
      ", ϕ0=" <> ToString[NumberForm[ϕStart // N, 3]] <>
      ", v0=" <> ToString[NumberForm[vStart // N, 3]] <>
      ", θ0=" <> ToString[NumberForm[θStart // N, 3]] <>
      ", γ=" <> ToString[NumberForm[viscosity // N, 3]] <>
      ", Λ=" <> ToString[NumberForm[spinAngle // N, 3]],
     FontFamily → "Cambria", FontSize → 18],
   GridLines → {{Δtmin, Δtmax}, None},
   Epilog → fitPlotEpilog
   (*Epilog→{fitPlotEpilog,Directive[lineStyle],line}*)],

  Show[tLogSepPlot,
   PlotLabel → Style[
     "Δt = " <> ToString[NumberForm[Δt // N, 3]] <>
      ", δ = " <> ToString[NumberForm[δ // N, 3]] <>
      ", r=" <> ToString[NumberForm[acRatio // N, 3]] <>
      ", Ω=" <> ToString[NumberForm[spinRate // N, 3]] <>
      ", λ0=" <> ToString[NumberForm[λStart // N, 4]] <>
      ", ϕ0=" <> ToString[NumberForm[ϕStart // N, 3]] <>
      ", v0=" <> ToString[NumberForm[vStart // N, 3]] <>
      ", θ0=" <> ToString[NumberForm[θStart // N, 3]] <>
      ", γ=" <> ToString[NumberForm[viscosity // N, 3]] <>
      ", Λ=" <> ToString[NumberForm[spinAngle // N, 3]],
     FontFamily → "Cambria", FontSize → 18],
   GridLines → {{Δtmin, Δtmax}, None}
   (*Epilog→fitPlotEpilog,*)
   (*Epilog→{Directive[lineStyle],line}*)]];

 trajectoryPlot =
  Show[surfacePlot, pathPlotF, pathPlotA, (*startPoint,*)axisPlot];

 GraphicsRow[{lyapunovPlot, trajectoryPlot}, ImageSize → 800, Spacings → 0]
]
```

In[353]:=
```
Manipulate[
 maxLyap[acRatio, Λ, ω, γ][logδLat, T, Δt, λ0, ϕ0, v, θ][Δtmin, Δtwidth][
    plotPoints, Tstart, Tend][showFit][zoom],
 {{acRatio, 5 / 4}, 0.5, 3}, {{ω, 1}, -10, 10}, {{Λ, 0}, 0, π / 2}, Delimiter,
 {{logδLat, -12}, -15, -1}, {{λ0, 0.01}, -1.5, 1.5}, {{ϕ0, 3.64}, 0 °, 360 °},
 {{γ, 0}, 0, 5}, {{v, 0}, 0, 5}, {{θ, 0}, 0, π}, {{λ0, 0.01}, -1.5, 1.5},
 Delimiter, {{T, 500}, 10, 2000}, {{Δt, 0.5}, 0.1, 2}, {{zoom, T}, 10, T},
 Delimiter, {{Δtmin, Δt}, Δt, Max[Δt, T - Δtwidth] },
 {{Δtwidth, T - Δtmin}, Δt, T - Δtmin},
 Delimiter, {{plotPoints, 40}, 10, 200, 1}, {{Tstart, 0}, 0, Tend},
 {{Tend, 100}, 0, T}, {{showFit, True}, {True, False}}]
```

2D Sliding Paths

Define a function to display a two dimensional projection of the slider's path

```
frame2DLines[acRatio_, spinRate_, spinAngle_, viscosity_][λStart_, ϕStart_,
    vStart_, θStart_][tMax_, dt0_, showMarks_, translucent_, size_][wp_] :=
 Module[{parameters, initials, pData, gData, nData, tooBig,
   λs, ϕs, tS, dt, ϕMod, λMod, pathPlot, startPlot},
  parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → viscosity};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};

  {λs, ϕs} = Check[NDSolveValue[IVPN2 /. parameters /. initials, {λN, ϕN},
     {tN, 0, tMax}, MaxSteps → Infinity, WorkingPrecision → wp], $Failed];

  pData = {}; tS = 0;
  While[tS ≤ tMax,
   ϕMod = Mod[ϕs[tS], 2 π];
   λMod = Mod[λs[tS], π, -π / 2];
   AppendTo[pData, {ϕMod, λMod}];
   dt = dt0 Cos[λMod]^1.2; (* nonlinear sampling *)
   tS += dt
  ];

  nData = Length@pData;
  tooBig = π;
  gData = Table[Which[
      pData[[n + 1, 1]] - pData[[n, 1]] < -tooBig, (* draw off right edge *)
      {Hue[0.85 n / nData], Line[{pData[[n]], pData[[n + 1]] + {2 π, 0}}],
       Line[{pData[[n]] - {2 π, 0}, pData[[n + 1]]}]},

      pData[[n + 1, 1]] - pData[[n, 1]] > tooBig, (* draw off left edge *)
```

```
      {Hue[0.85 n / nData], Line[{pData[[n]], pData[[n + 1]] - {2 π, 0}}],
       Line[{pData[[n]] + {2 π, 0}, pData[[n + 1]]}]},

      True, {Hue[0.85 n / nData], Line[{pData[[n]], pData[[n + 1]]}]}
      (* draw through interior *)
     ], {n, 1, nData - 1}] // Flatten;

pathPlot = Graphics[If[translucent, Prepend[gData, Opacity[0.5]], gData]];

startPlot = Graphics[If[showMarks,
    {Red, Disk[{ϕStart, λStart}, 0.03], Black, Circle[{0, spinAngle}, 0.06]}]
  , PlotRange → {{0, 2 π}, {-π / 2, π / 2}}
  , Frame → True
  , FrameLabel → {Style[ϕ, FontFamily → "Cambria", FontSize → 20],
     Style[λ, FontFamily → "Cambria", FontSize → 20]}
  , FrameTicks → {{Range[-π / 2, π / 2, π / 2], None}, {Range[0, 2 π, π], None}}
  , AspectRatio → Automatic (* acRatio *)
  , ImageSize → size
  , PlotRangeClipping → True
  , PlotLabel → Style[
     "λ0=" <> ToString[NumberForm[λStart // N, 3]] <>
      ", ϕ0=" <> ToString[NumberForm[ϕStart // N, 3]] <>
      ", r=" <> ToString[NumberForm[acRatio // N, 3]] <>
      ", Ω=" <> ToString[NumberForm[spinRate // N, 3]] <>
      ", v0=" <> ToString[NumberForm[vStart // N, 3]] <>
      ", θ0=" <> ToString[NumberForm[θStart // N, 3]] <>
      ", γ=" <> ToString[NumberForm[viscosity // N, 3]] <>
      ", T=" <> ToString[tMax] <>
      ", Λ=" <> ToString[NumberForm[spinAngle // N, 3]],
     FontFamily → "Cambria", FontSize → 18],
   LabelStyle → Directive[FontFamily → "Cambria", FontSize → 20]
  ];

If[λs == $Failed ⋁ ϕs == $Failed, Show[startPlot],
 Show[startPlot, pathPlot], Show[startPlot, pathPlot]]
]
```

```
With[{δ = 10^-9, δ2 = 5 × 10^-2},
 Manipulate[frame2DLines[rN, ωSpinN, λSpinN, γN][λ0, ϕ0, v0, θ0][
    tMax, dt0, showMarks, translucent, Large]
  , {{rN, 0.8, "r_n"}, δ2, 3}
  , {{ωSpinN, 2.25, "Ω_n"}, -7, 7}
  , {{λSpinN, 0, "Λ_n"}, 0, π/2}
  , {{γN, 0.07, "γ_n"}, 0, 1}
  , Delimiter
  , {{λ0, -1, "λ_n0"}, -π/2 + 0.25, π/2 - δ}
  , {{ϕ0, 0, "ϕ_n0"}, 0, 2 π}
  , {{v0, 0, "V_n0"}, 0, 9}
  , {{θ0, 1, "θ_n0"}, δ, 2 π - δ}
  , Delimiter
  , {{tMax, 10, "max t_n"}, 1, 100}
  , {{dt0, 0.1, "Δt_0"}, 0.005, 0.5}
  , {{showMarks, True, "marks"}, {True, False}}
  , {{translucent, False, "translucent"}, {True, False}}
 ]
]
```

Basin Boundaries

Function for latitude attractors

```
frame2DBasinsλ[acRatio_, spinRate_, spinAngle_, viscosity_][
   λStart_, ϕStart_, vStart_, θStart_][tMax_] :=
 Module[{parameters, initials, λs, ϕs, basinID, IVNP2},
  parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → viscosity};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};

  {λs, ϕs} = Check[NDSolveValue[IVPN2 /. parameters /. initials,
     {λN, ϕN}, {tN, 0, tMax}, MaxSteps → Infinity], $Failed];

  basinID = If[λs[tMax] < 0, 1, 0];

  If[λs == $Failed ⋁ ϕs == $Failed, -1, basinID, basinID]
 ]
```

Function for longitude attractors

```
frame2DBasinsϕ[acRatio_, spinRate_, spinAngle_, viscosity_][
   λStart_, ϕStart_, vStart_, θStart_][tMax_] :=
 Module[{parameters, initials, λs, ϕs, basinID, IVNP2},
  parameters = {rN → acRatio, ωSpinN → spinRate, λSpinN → spinAngle, γN → viscosity};
  initials = {λ0 → λStart, ϕ0 → ϕStart, v0N → vStart, θ0 → θStart};

  {λs, ϕs} = Check[NDSolveValue[IVPN2 /. parameters /. initials,
     {λN, ϕN}, {tN, 0, tMax}, MaxSteps → Infinity], $Failed];

  basinID = If[λs[tMax] < 0, 1, 0];


  basinID := -2 /; λs[tMax] > -3 π / 7;
  basinID := 2 /; λs[tMax] < 3 π / 7;
  basinID := 1 /; ϕs[tMax] < 0.2;
  basinID := -1 /; ϕs[tMax] > 6.1;


  If[λs == $Failed ⋁ ϕs == $Failed, 0, basinID, basinID]
 ]
```

Project basin plot onto surface of the spheroid

```
λs = Table[λ, {λ, -0.999 π/2, π/2, δ}];

ϕs = Table[ϕ, {ϕ, 0.001, 2 π, δ}];

points = Flatten[Outer[List, λs, ϕs], 1];

colors = Flatten@basins1 /. {-1 → White, 0 → Green, 1 → Purple};

positions =
  Point[{0.85 Cos[#[[1]]] Cos[#[[2]]], 0.85 Cos[#[[1]]] Sin[#[[2]]], Sin[#[[1]]]}] & /@ points;

list = Transpose[{colors, positions}];

Graphics3D[list, Boxed → False]
```

Divergence Plots

Difference Function

Get the difference between fiducial and adjacent trajectories at time T

```
difference[acRatio_, ωω_, vStart_, θStart_][
  δlat0_, δlong0_, λStart_, ϕStart_, T_] := Module[
  {parameters, fStart, fiducialOrbit, aStart, adjacentOrbit, δOrbit},

  parameters = {
    rN → acRatio,
    ωSpinN → ωω,
    γN → 0
   };

  initialsf = {λ0 → λStart, ϕ0 → ϕStart, v0 → vStart, θ0 → θStart};
  fiducialOrbit[tt_] = {λf, ϕf} = NDSolveValue[IVPN2 /. parameters /. initialsf,
      {λN, ϕN}, {tN, 0, T}, MaxSteps → Infinity] /. tN → tt;
  initialsa = {λ0 → λStart + δlat0, ϕ0 → ϕStart, v0 → vStart, θ0 → θStart};
  adjacentOrbit[tt_] = {λa, ϕa} = NDSolveValue[IVPN2 /. parameters /. initialsa,
      {λN, ϕN}, {tN, 0, T}, MaxSteps → Infinity] /. tN → tt;

  δOrbit[t_] = {λf[t], ϕf[t]} - {λa[t], ϕa[t]};
  Norm@δOrbit[T]
 ]
```

Divergence plots

```
ellipsoidSelect[acRatio_, ωx_, vStart_, θStart_][δlat_, δlong_, δlat0_, T_][
  cutoff_] := Module[
  {(*points,*)(*minδ,maxδ,egg,parameters*)},
  parameters = {
    rN → acRatio,
    ωSpinN → ωx,
    γN → 0
   };

  rawpoints = ParallelTable[{{lat0, long0},
     difference[acRatio, ωx, vStart, θStart][δlat0, δlat0, lat0, long0, T]},
    {lat0, -1.3, 1.3, δlat}, {long0, 0.01, 2 π - 0.01, δlong}];
  points = Partition[Flatten[rawpoints, 2], 2];

  data = Reverse[
    points[[Select[Range[Length@points], points[[#, 2]] > 10^cutoff &], 1]], 2];

  egg = ListPlot[data,
    PlotLabel → Style[
      "r=" <> ToString[NumberForm[acRatio // N, 3]] <>
       ", ω=" <> ToString[NumberForm[ωx // N, 3]] <>
       ", Λ=" <> ToString[NumberForm[0 // N, 3]] <>
       ", v0=" <> ToString[NumberForm[vStart // N, 3]] <>
       ", θ0=" <> ToString[NumberForm[θStart // N, 3]]
      , FontFamily → "Cambria", FontSize → 18], Axes → False, Frame → True,
    FrameTicks → {{{-1.5, 0, 1.5}, None}, {{0, Pi, 2 Pi}, None}}, FrameLabel → {ϕ, λ},
    LabelStyle → Directive[FontSize → 20, FontFamily → "Cambria"],
    PlotRange → {{0, 2 π}, {-π/2, π/2}}, PlotStyle → Directive[PointSize → 0.006]]
  ;

  Show[egg]
 ]
```

Bibliography

[1] Meech et al. Discovery and characterization of the first known interstellar object. *Nature*, 2017.

[2] ESO/K. Meech et al. Light curve of interstellar asteroid ‘oumuamua.

[3] N. Moore and J. F. Lindner. Sliding on a spinning asteroid (geodesics on a rotating ellipsoid). In *APS March Meeting Abstracts*, 2017.

[4] William Duncan MacMillan. *The Theory of the Potential*, chapter The Newtonian Potential Function, pages 45–49. McGraw-Hill Book Company, inc., New York, 1 edition, 1930.

[5] John Taylor. *Classical Mechanics*. 5 edition, 2005.

[6] Hannah Peltz Smalley. Visualizing the complex behavior of a pendulum. 5 2017.

[7] T.S Parker & L.O. Chua. *Practical Algorithms for Chaotic Systems*. 1 edition, 1989.

[8] C.E. Meador. Numerical calculation of lyapunov exponents for three-dimensional systems of ordinary differential equations. *Theses, Dissertations and Capstones*, 2011.

[9] Garfinkel et al. Controlling cardiac chaos. *Science*, 1992.

[10] Vladimir Aslanov and Vadim Yudintsev. Dynamics and chaos control of gyrostat satellite. *Chaos, Solitons Fractals*, 45(9):1100 – 1107, 2012.